RIFERIMENTO al Linguaggio MQL5

per il terminale client MetaTrader 5

Studia l' MQL5 e risolvi ogni attività:

- Crea i tuoi indicatori di analisi tecnica di qualunque complessità
- Usa l'autotrading: sistemi di trading automatizzato per lavorare in vari mercati finanziari
- Sviluppa i tuoi utensili analitici sui successi matematici e sui metodi tradizionali
- Scrivi sistemi informativi di trading per risolvere una varia gamma di attività (trading, monitoraggio, allarme, ecc.)

Content

MQL5 Riferimento 71

1 Le basi del linguaggio ... 73

Sintassi ... 74
- Commenti ... 75
- Identificatori ... 76
- Parole riservate ... 77

Tipi di Dati ... 79
- Tipi Integer ... 80
  - Tipi Char, Short, Int e Long ... 81
  - Costanti Carattere ... 85
  - Tipo Datetime ... 89
  - Tipo Colore ... 90
  - Tipo Bool ... 91
  - Enumerazioni ... 92
- Tipi Reali (double, float) ... 94
- Numero complesso (complex) ... 100
- Tipo Stringa ... 102
- Strutture, Classi e Interfacce ... 103
- Oggetto Array Dinamico ... 130
- Matrices and vectors ... 131
- Typecasting ... 138
- Tipo Void e Costante NULL ... 143
- Tipi definiti dall'utente ... 144
- Puntatori agli Oggetti ... 154
- Riferimento. Il Modifier & e la parola chiave this ... 158

Operazioni ed Espressioni ... 160
- Espressioni ... 161
- Operazioni Aritmetiche ... 162
- Operazioni di Assegnazione ... 163
- Operazioni di Assegnazione ... 164
- Operazioni Booleani ... 165
- Operazioni bit per bit ... 167
- Altre Operazioni ... 170
- Regole di Precedenza ... 174

Operatori ... 176
- Operatore Composto ... 178
- Operatore Espressione ... 179
- Operatore Return ... 180
- Operatore Condizionale if-else ... 181
- Operatore Ternario ?: ... 182
- Operatore Switch ... 184
- Operatore Ciclico While ... 186
- Operatore Ciclico for ... 187
- Operatore Ciclico do while ... 189
- Operatore Break ... 190
- Operatore Continue ... 191
- Operatore new di Creazione Oggetto ... 192
- Operatore Cancellatore Oggetto delete ... 194

Funzioni ... 195
- Chiamate di funzione ... 197
- Passaggio di Parametri ... 198
- L'overloading di funzioni ... 201
- Operazione Overloading ... 204

Descrizione delle Funzioni Esterne ... 218
Esportazione di funzioni ... 220
Funzioni di Gestione degli Eventi ... 221
Variabili ... 233
Variabili locali ... 237
Parametri Formali ... 239
Variabili Statiche ... 241
Variabili Globali ... 243
Variabili di Input ... 244
Variabili Esterne ... 251
Inizializzazione delle Variabili ... 252
Visibilità Ambito(_*scope) e Durata delle Variabili(_*lifetime) ... 254
Creazione ed Eliminazione di oggetti ... 256
Preprocessore ... 259
Sostituzione Macro (#define) ... 261
Proprietà del programma (#property) ... 264
Includere Files (# include) ... 270
Funzioni di Importazione (#import) ... 271
Conditional Compilation (#ifdef, #ifndef, #else, #endif) ... 274
Programmazione Ad Oggetti ... 276
Incapsulamento ed estensibilità dei Tipi ... 278
Ereditarietà ... 281
Polimorfismo ... 286
L' Overload ... 290
Funzioni Virtuali ... 291
I Membri Statici di una classe ... 295
Templates di Funzioni ... 299
Templates delle Classi ... 303
Classi Astratte ... 308
Namespaces ... 310
2 Costanti, Enumerazioni e Strutture ... 313
Costanti del Grafico ... 314
Tipi di Eventi del Grafico ... 315
Timeframes del Grafico ... 323
Proprietà Grafico ... 325
Posizionamento Costanti ... 333
Rappresentazione Grafico ... 334
Esempi di Lavoro con il grafico Chart ... 336
Costanti Oggetti ... 394
Tipi di oggetti ... 395
OBJ_VLINE ... 397
OBJ_HLINE ... 402
OBJ_TREND ... 407
OBJ_TRENDBYANGLE ... 414
OBJ_CYCLES ... 420
OBJ_ARROWED_LINE ... 426
OBJ_CHANNEL ... 432
OBJ_STDDEVCHANNEL ... 439
OBJ_REGRESSION ... 446
OBJ_PITCHFORK ... 452
OBJ_GANNLINE ... 460
OBJ_GANNFAN ... 467
OBJ_GANNGRID ... 474
OBJ_FIBO ... 481
OBJ_FIBOTIMES ... 488
OBJ_FIBOFAN ... 495
OBJ_FIBOARC ... 502
OBJ_FIBOCHANNEL ... 509

OBJ_EXPANSION ... 517
OBJ_ELLIOTWAVE5 ... 525
OBJ_ELLIOTWAVE3 ... 533
OBJ_RECTANGLE ... 540
OBJ_TRIANGLE ... 546
OBJ_ELLIPSE ... 553
OBJ_ARROW_THUMB_UP ... 560
OBJ_ARROW_THUMB_DOWN ... 566
OBJ_ARROW_UP ... 572
OBJ_ARROW_DOWN ... 578
OBJ_ARROW_STOP ... 584
OBJ_ARROW_CHECK ... 590
OBJ_ARROW_LEFT_PRICE ... 596
OBJ_ARROW_RIGHT_PRICE ... 601
OBJ_ARROW_BUY ... 606
OBJ_ARROW_SELL ... 611
OBJ_ARROW ... 616
OBJ_TEXT ... 622
OBJ_LABEL ... 628
OBJ_BUTTON ... 636
OBJ_CHART ... 643
OBJ_BITMAP ... 650
OBJ_BITMAP_LABEL ... 657
OBJ_EDIT ... 664
OBJ_EVENT ... 671
OBJ_RECTANGLE_LABEL ... 676
Object Properties ... 682
Metodi di Binding Ogetti ... 711
Angolo del Grafico ... 716
Visibilità degli oggetti ... 719
Livelli delle Elliott Wave (_* Onde di Elliot) ... 722
Oggetti Gann ... 723
I Web Colors ... 725
Wingdings ... 727
Costanti Indicatore ... 728
Costanti Prezzo ... 729
Metodi di smussamento ... 732
Linee Indicatori ... 733
Stili di Disegno ... 735
Proprietà Indicatore Personalizzato ... 741
Tipi di indicatori ... 747
Identificatori Tipo di Dati ... 749
Stato dell'Ambiente ... 750
Proprietà del Terminale Client ... 751
Proprietà dell' Esecuzione di Programmi MQL5 ... 757
Proprietà dei Simboli ... 761
Proprietà Account ... 845
Statistiche Testing ... 854
Costanti di Trade ... 859
Proprietà del database della CroniStoria ... 860
Proprietà Ordini ... 861
Proprietà Posizione ... 880
Proprietà Affari(Deal) ... 885
Tipi di Operazioni di Trade ... 889
Tipi di Transazioni di Trade ... 901
Gli ordini di Trade nel DOM ... 904
Proprietà dei Segnali ... 905
Costanti Nominate ... 907

Sostituzioni Macro predefinite ... 908
Costanti Matematiche ... 914
Costanti di tipo numerico ... 916
Codici del Motivo di DeInizializzazione ... 919
Controllo del Puntatore Oggetti ... 921
Altre costanti ... 922
Strutture Dati ... 926
Struttura del Tipo Data ... 927
Struttura dei parametri di Input ... 928
Struttura Dati dello Storico ... 929
Struttura del Depth of Market ... 930
Struttura Richiesta di Trade ... 931
Struttura di Risultati di Richiesta Controllo ... 944
Struttura di un Risultato di Richiesta di Trade ... 945
Struttura di una Transazione di Trade ... 948
Struttura per i Prezzi Correnti ... 955
Economic Calendar structures ... 957
I codici di Errore e di Avvertenza ... 962
Codici di Ritorno del Trade Server ... 963
Avvisi del compilatore ... 966
Errori di compilazione ... 969
Errori di Runtime ... 980
Costanti Input/Output ... 994
Flags di Apertura File ... 995
Proprietà File ... 997
Posizione In-File ... 998
L'uso di una tabella Codici ... 999
MessageBox ... 1000
3 Programmi MQL5 ... 1002
Esecuzione dei Programmi ... 1003
Trade Permission ... 1010
Eventi Terminale Client ... 1014
Risorse ... 1018
Chiamata delle funzioni importate ... 1030
Errori di Runtime ... 1032
Testare Strategie di Trading ... 1033
4 Variabili predefinite ... 1060
_AppliedTo ... 1061
_Digits ... 1063
_Point ... 1064
_LastError ... 1065
_Period ... 1066
_RandomSeed ... 1067
_StopFlag ... 1068
_Symbol ... 1069
_UninitReason ... 1070
_IsX64 ... 1071
5 Funzioni Comuni ... 1072
Alert ... 1074
CheckPointer ... 1075
Comment ... 1077
CryptEncode ... 1079
CryptDecode ... 1081
DebugBreak ... 1082
ExpertRemove ... 1083
GetPointer ... 1085
GetTickCount ... 1089

GetTickCount64 ... 1090
GetMicrosecondCount ... 1091
MessageBox ... 1093
PeriodSeconds ... 1094
PlaySound ... 1095
Print ... 1096
PrintFormat ... 1098
ResetLastError ... 1104
ResourceCreate ... 1105
ResourceFree ... 1107
ResourceReadImage ... 1108
ResourceSave ... 1109
SetReturnError ... 1110
SetUserError ... 1111
Sleep ... 1112
TerminalClose ... 1113
TesterStatistics ... 1115
TesterStop ... 1116
TesterDeposit ... 1117
TesterHideIndicators ... 1118
TesterWithdrawal ... 1120
TranslateKey ... 1121
ZeroMemory ... 1122

6 Funzioni di Array ... 1123

ArrayBsearch ... 1124
ArrayCopy ... 1128
ArrayCompare ... 1133
ArrayFree ... 1134
ArrayGetAsSeries ... 1143
ArrayInitialize ... 1146
ArrayFill ... 1148
ArrayIsDynamic ... 1150
ArrayIsSeries ... 1152
ArrayMaximum ... 1154
ArrayMinimum ... 1165
ArrayPrint ... 1176
ArrayRange ... 1179
ArrayResize ... 1180
ArrayInsert ... 1183
ArrayRemove ... 1186
ArrayReverse ... 1188
ArraySetAsSeries ... 1190
ArraySize ... 1193
ArraySort ... 1195
ArraySwap ... 1200

7 Metodi Matriciali e Vettoriali ... 1202

Tipi di matrice e vettore ... 1209
Enumerazioni ... 1210
Initialization ... 1215
Assign ... 1218
CopyIndicatorBuffer ... 1220
CopyRates ... 1222
CopyTicks ... 1226
CopyTicksRange ... 1229
Eye ... 1231
Identity ... 1233
Ones ... 1235
Zeros ... 1236

Full ... 1237
Tri ... 1238
Init ... 1239
Fill ... 1241
Manipolazioni ... 1242
HasNan ... 1244
Transpose ... 1245
TriL ... 1247
TriU ... 1248
Diag ... 1249
Row ... 1251
Col ... 1253
Copy ... 1255
Compare ... 1257
CompareByDigits ... 1259
Flat ... 1261
Clip ... 1263
Reshape ... 1264
Resize ... 1266
Set ... 1268
SwapRows ... 1270
SwapCols ... 1271
Split ... 1272
Hsplit ... 1274
Vsplit ... 1276
ArgSort ... 1278
Sort ... 1279
Operazioni ... 1281
Operazioni matematiche ... 1283
Funzioni matematiche ... 1284
Prodotti ... 1285
MatMul ... 1286
GeMM ... 1291
Power ... 1295
Dot ... 1298
Kron ... 1300
Inner ... 1302
Outer ... 1304
CorrCoef ... 1307
Cov ... 1311
Correlate ... 1314
Convolve ... 1317
Trasformazioni ... 1320
Cholesky ... 1321
Eig ... 1322
EigVals ... 1325
LU ... 1326
LUP ... 1328
QR ... 1330
SVD ... 1332
Statistica ... 1335
ArgMax ... 1336
ArgMin ... 1337
Max ... 1338
Min ... 1339
Ptp ... 1340
Sum ... 1341
Prod ... 1342

CumSum ... 1344
CumProd ... 1346
Percentile ... 1348
Quantile ... 1350
Median ... 1352
Mean ... 1354
Average ... 1355
Std ... 1357
Var ... 1359
LinearRegression ... 1361
Features ... 1364
Rows ... 1365
Cols ... 1366
Size ... 1367
Norm ... 1368
Cond ... 1371
Det ... 1374
SLogDet ... 1376
Rank ... 1377
Trace ... 1380
Spectrum ... 1381
Soluzioni ... 1382
Solve ... 1383
LstSq ... 1384
Inv ... 1385
PInv ... 1387
Machine learning ... 1388
Activation ... 1393
Derivative ... 1397
Loss ... 1399
LossGradient ... 1401
RegressionMetric ... 1403
ConfusionMatrix ... 1405
ConfusionMatrixMultilabel ... 1407
ClassificationMetric ... 1409
ClassificationScore ... 1412
PrecisionRecall ... 1416
ReceiverOperatingCharacteristic ... 1420
8 Funzioni di Conversione ... 1424
CharToString ... 1426
CharArrayToString ... 1427
CharArrayToStruct ... 1428
StructToCharArray ... 1429
ColorToARGB ... 1430
ColorToString ... 1432
DoubleToString ... 1433
EnumToString ... 1434
IntegerToString ... 1436
ShortToString ... 1437
ShortArrayToString ... 1438
TimeToString ... 1439
NormalizeDouble ... 1440
StringToCharArray ... 1442
StringToColor ... 1443
StringToDouble ... 1444
StringToInteger ... 1445
StringToShortArray ... 1446
StringToTime ... 1447

StringFormat 1448
9 Funzioni Matematiche 1452
MathAbs 1454
MathArccos 1455
MathArcsin 1456
MathArctan 1457
MathArctan2 1458
MathClassify 1459
MathCeil 1461
MathCos 1462
MathExp 1463
MathFloor 1464
MathLog 1465
MathLog10 1466
MathMax 1467
MathMin 1468
MathMod 1469
MathPow 1470
MathRand 1471
MathRound 1472
MathSin 1473
MathSqrt 1474
MathSrand 1475
MathTan 1478
MathIsValidNumber 1479
MathExpm1 1480
MathLog1p 1481
MathArccosh 1482
MathArcsinh 1483
MathArctanh 1484
MathCosh 1485
MathSinh 1486
MathTanh 1487
MathSwap 1488
10 Funzioni Stringa 1489
StringAdd 1490
StringBufferLen 1492
StringCompare 1493
StringConcatenate 1495
StringFill 1496
StringFind 1497
StringGetCharacter 1498
StringInit 1499
StringLen 1500
StringSetLength 1501
StringReplace 1502
StringReserve 1503
StringSetCharacter 1505
StringSplit 1507
StringSubstr 1509
StringToLower 1510
StringToUpper 1511
StringTrimLeft 1512
StringTrimRight 1513
11 Data ed Ora 1514
TimeCurrent 1515
TimeTradeServer 1516

TimeLocal 1517
TimeGMT 1518
TimeDaylightSavings 1519
TimeGMTOffset 1520
TimeToStruct 1521
StructToTime 1522
12 Informazioni account 1523
AccountInfoDouble 1524
AccountInfoInteger 1525
AccountInfoString 1527
13 Verifica Stato 1528
GetLastError 1529
IsStopped 1530
UninitializeReason 1531
TerminalInfoInteger 1532
TerminalInfoDouble 1533
TerminalInfoString 1534
MQLInfoInteger 1535
MQLInfoString 1536
Symbol 1537
Period 1538
Digits 1539
Point 1540
14 Gestione degli Eventi 1541
OnStart 1543
OnInit 1546
OnDeinit 1549
OnTick 1552
OnCalculate 1558
OnTimer 1562
OnTrade 1565
OnTradeTransaction 1570
OnBookEvent 1576
OnChartEvent 1579
OnTester 1586
OnTesterInit 1593
OnTesterDeinit 1600
OnTesterPass 1601
15 Market Info 1602
SymbolsTotal 1603
SymbolExist 1604
SymbolName 1605
SymbolSelect 1606
SymbolIsSynchronized 1607
SymbolInfoDouble 1608
SymbolInfoInteger 1610
SymbolInfoString 1612
SymbolInfoMarginRate 1614
SymbolInfoTick 1615
SymbolInfoSessionQuote 1616
SymbolInfoSessionTrade 1617
MarketBookAdd 1618
MarketBookRelease 1619
MarketBookGet 1620
16 Calendario Economico 1621
CalendarCountryById 1622

CalendarEventById 1624
CalendarValueById 1627
CalendarCountries 1630
CalendarEventByCountry 1632
CalendarEventByCurrency 1634
CalendarValueHistoryByEvent 1636
CalendarValueHistory 1639
CalendarValueLastByEvent 1642
CalendarValueLast 1647
17 Accesso alle Timeseries ed Indicatori 1652
Direzione di Indicizzazione negli Array, Buffers e TimeSeries 1657
Organizzazione di Accesso ai Dati 1661
SeriesInfoInteger 1671
Bars 1673
BarsCalculated 1676
IndicatorCreate 1678
IndicatorParameters 1681
IndicatorRelease 1683
CopyBuffer 1685
CopyRates 1690
CopySeries 1694
CopyTime 1698
CopyOpen 1701
CopyHigh 1704
CopyLow 1708
CopyClose 1711
CopyTickVolume 1714
CopyRealVolume 1718
CopySpread 1721
CopyTicks 1725
CopyTicksRange 1731
iBars 1733
iBarShift 1734
iClose 1737
iHigh 1739
iHighest 1741
iLow 1742
iLowest 1744
iOpen 1745
iTime 1747
iTickVolume 1749
iRealVolume 1751
iVolume 1753
iSpread 1755
18 Simboli Personalizzati 1757
CustomSymbolCreate 1759
CustomSymbolDelete 1761
CustomSymbolSetInteger 1762
CustomSymbolSetDouble 1763
CustomSymbolSetString 1764
CustomSymbolSetMarginRate 1765
CustomSymbolSetSessionQuote 1766
CustomSymbolSetSessionTrade 1767
CustomRatesDelete 1768
CustomRatesReplace 1769
CustomRatesUpdate 1770
CustomTicksAdd 1771
CustomTicksDelete 1773

CustomTicksReplace 1774
CustomBookAdd 1776
19 Operazioni col Grafico 1779
ChartApplyTemplate 1781
ChartSaveTemplate 1784
ChartWindowFind 1789
ChartTimePriceToXY 1791
ChartXYToTimePrice 1792
ChartOpen 1794
ChartFirst 1795
ChartNext 1796
ChartClose 1797
ChartSymbol 1798
ChartPeriod 1799
ChartRedraw 1800
ChartSetDouble 1801
ChartSetInteger 1802
ChartSetString 1804
ChartGetDouble 1806
ChartGetInteger 1808
ChartGetString 1810
ChartNavigate 1812
ChartID 1815
ChartIndicatorAdd 1816
ChartIndicatorDelete 1820
ChartIndicatorGet 1823
ChartIndicatorName 1825
ChartIndicatorsTotal 1826
ChartWindowOnDropped 1827
ChartPriceOnDropped 1828
ChartTimeOnDropped 1829
ChartXOnDropped 1830
ChartYOnDropped 1831
ChartSetSymbolPeriod 1832
ChartScreenShot 1833
20 Funzioni di Trade 1836
OrderCalcMargin 1838
OrderCalcProfit 1839
OrderCheck 1840
OrderSend 1841
OrderSendAsync 1846
PositionsTotal 1857
PositionGetSymbol 1858
PositionSelect 1859
PositionSelectByTicket 1860
PositionGetDouble 1861
PositionGetInteger 1862
PositionGetString 1864
PositionGetTicket 1865
OrdersTotal 1866
OrderGetTicket 1867
OrderSelect 1869
OrderGetDouble 1870
OrderGetInteger 1871
OrderGetString 1872
HistorySelect 1873
HistorySelectByPosition 1875
HistoryOrderSelect 1876

HistoryOrdersTotal ... 1877
HistoryOrderGetTicket ... 1878
HistoryOrderGetDouble ... 1880
HistoryOrderGetInteger ... 1881
HistoryOrderGetString ... 1884
HistoryDealSelect ... 1885
HistoryDealsTotal ... 1886
HistoryDealGetTicket ... 1887
HistoryDealGetDouble ... 1890
HistoryDealGetInteger ... 1891
HistoryDealGetString ... 1894
21 Segnali di Trade ... 1895
SignalBaseGetDouble ... 1896
SignalBaseGetInteger ... 1897
SignalBaseGetString ... 1898
SignalBaseSelect ... 1899
SignalBaseTotal ... 1900
SignalInfoGetDouble ... 1901
SignalInfoGetInteger ... 1902
SignalInfoGetString ... 1903
SignalInfoSetDouble ... 1904
SignalInfoSetInteger ... 1905
SignalSubscribe ... 1906
SignalUnsubscribe ... 1907
22 Funzioni di Rete ... 1908
SocketCreate ... 1910
SocketClose ... 1913
SocketConnect ... 1916
SocketIsConnected ... 1920
SocketIsReadable ... 1921
SocketIsWritable ... 1924
SocketTimeouts ... 1925
SocketRead ... 1926
SocketSend ... 1930
SocketTlsHandshake ... 1934
SocketTlsCertificate ... 1935
SocketTlsRead ... 1939
SocketTlsReadAvailable ... 1943
SocketTlsSend ... 1944
WebRequest ... 1945
SendFTP ... 1948
SendMail ... 1949
SendNotification ... 1950
23 Variabili Globali del Terminale ... 1951
GlobalVariableCheck ... 1952
GlobalVariableTime ... 1953
GlobalVariableDel ... 1954
GlobalVariableGet ... 1955
GlobalVariableName ... 1956
GlobalVariableSet ... 1957
GlobalVariablesFlush ... 1958
GlobalVariableTemp ... 1959
GlobalVariableSetOnCondition ... 1960
GlobalVariablesDeleteAll ... 1961
GlobalVariablesTotal ... 1962
24 Funzioni con i File ... 1963
FileSelectDialog ... 1966

FileFindFirst 1969
FileFindNext 1971
FileFindClose 1973
FileIsExist 1975
FileOpen 1978
FileClose 1981
FileCopy 1982
FileDelete 1985
FileMove 1987
FileFlush 1989
FileGetInteger 1991
FileIsEnding 1994
FileIsLineEnding 1996
FileReadArray 2001
FileReadBool 2003
FileReadDatetime 2006
FileReadDouble 2009
FileReadFloat 2012
FileReadInteger 2015
FileReadLong 2019
FileReadNumber 2022
FileReadString 2027
FileReadStruct 2029
FileSeek 2033
FileSize 2036
FileTell 2038
FileWrite 2041
FileWriteArray 2044
FileWriteDouble 2047
FileWriteFloat 2050
FileWriteInteger 2052
FileWriteLong 2055
FileWriteString 2057
FileWriteStruct 2060
FileLoad 2063
FileSave 2065
FolderCreate 2067
FolderDelete 2070
FolderClean 2073

25 Indicatori Personalizzati 2076

Stili Indicatore negli Esempi 2080
DRAW_NONE 2091
DRAW_LINE 2094
DRAW_SECTION 2098
DRAW_HISTOGRAM 2102
DRAW_HISTOGRAM2 2106
DRAW_ARROW 2110
DRAW_ZIGZAG 2115
DRAW_FILLING 2120
DRAW_BARS 2125
DRAW_CANDLES 2131
DRAW_COLOR_LINE 2138
DRAW_COLOR_SECTION 2143
DRAW_COLOR_HISTOGRAM 2149
DRAW_COLOR_HISTOGRAM2 2154
DRAW_COLOR_ARROW 2159
DRAW_COLOR_ZIGZAG 2165
DRAW_COLOR_BARS 2170

DRAW_COLOR_CANDLES ... 2177
Connection between Indicator Properties and Functions ... 2184
SetIndexBuffer ... 2187
IndicatorSetDouble ... 2190
IndicatorSetInteger ... 2194
IndicatorSetString ... 2198
PlotIndexSetDouble ... 2201
PlotIndexSetInteger ... 2202
PlotIndexSetString ... 2206
PlotIndexGetInteger ... 2207
26 Oggetti Grafici ... 2210
ObjectCreate ... 2212
ObjectName ... 2216
ObjectDelete ... 2217
ObjectsDeleteAll ... 2218
ObjectFind ... 2219
ObjectGetTimeByValue ... 2220
ObjectGetValueByTime ... 2221
ObjectMove ... 2222
ObjectsTotal ... 2223
ObjectSetDouble ... 2224
ObjectSetInteger ... 2228
ObjectSetString ... 2231
ObjectGetDouble ... 2233
ObjectGetInteger ... 2235
ObjectGetString ... 2237
TextSetFont ... 2239
TextOut ... 2242
TextGetSize ... 2246
27 Indicatori Tecnici ... 2247
iAC ... 2250
iAD ... 2255
iADX ... 2260
iADXWilder ... 2265
iAlligator ... 2270
iAMA ... 2277
iAO ... 2282
iATR ... 2287
iBearsPower ... 2292
iBands ... 2297
iBullsPower ... 2303
iCCI ... 2308
iChaikin ... 2313
iCustom ... 2318
iDEMA ... 2322
iDeMarker ... 2327
iEnvelopes ... 2332
iForce ... 2338
iFractals ... 2343
iFrAMA ... 2348
iGator ... 2353
iIchimoku ... 2360
iBWMFI ... 2367
iMomentum ... 2372
iMFI ... 2377
iMA ... 2382
iOsMA ... 2387
iMACD ... 2392

iOBV 2398
iSAR 2403
iRSI 2408
iRVI 2413
iStdDev 2418
iStochastic 2423
iTEMA 2429
iTriX 2434
iWPR 2439
iVIDyA 2444
iVolumes 2449

28 Lavorare con i risultati di ottimizzazione 2454

FrameFirst 2456
FrameFilter 2457
FrameNext 2458
FrameInputs 2459
FrameAdd 2460
ParameterGetRange 2461
ParameterSetRange 2464

29 Funzioni Eventi 2466

EventSetMillisecondTimer 2467
EventSetTimer 2468
EventKillTimer 2469
EventChartCustom 2470

30 Lavorare con OpenCL 2476

CLHandleType 2478
CLGetInfoInteger 2479
CLGetInfoString 2482
CLContextCreate 2485
CLContextFree 2486
CLGetDeviceInfo 2487
CLProgramCreate 2492
CLProgramFree 2496
CLKernelCreate 2497
CLKernelFree 2498
CLSetKernelArg 2499
CLSetKernelArgMem 2500
CLSetKernelArgMemLocal 2501
CLBufferCreate 2502
CLBufferFree 2503
CLBufferWrite 2504
CLBufferRead 2509
CLExecute 2513
CLExecutionStatus 2515

31 Lavorare con i database 2516

DatabaseOpen 2520
DatabaseClose 2522
DatabaseImport 2523
DatabaseExport 2526
DatabasePrint 2532
DatabaseTableExists 2537
DatabaseExecute 2538
DatabasePrepare 2550
DatabaseReset 2559
DatabaseBind 2565
DatabaseBindArray 2570
DatabaseRead 2575

DatabaseReadBind 2576
DatabaseFinalize 2580
DatabaseTransactionBegin 2581
DatabaseTransactionCommit 2586
DatabaseTransactionRollback 2587
DatabaseColumnsCount 2588
DatabaseColumnName 2589
DatabaseColumnType 2590
DatabaseColumnSize 2591
DatabaseColumnText 2592
DatabaseColumnInteger 2593
DatabaseColumnLong 2594
DatabaseColumnDouble 2595
DatabaseColumnBlob 2596

32 Lavorare con DirectX 2597

DXContextCreate 2599
DXContextSetSize 2600
DXContextGetSize 2601
DXContextClearColors 2602
DXContextClearDepth 2603
DXContextGetColors 2604
DXContextGetDepth 2605
DXBufferCreate 2606
DXTextureCreate 2607
DXInputCreate 2613
DXInputSet 2614
DXShaderCreate 2615
DXShaderSetLayout 2616
DXShaderInputsSet 2617
DXShaderTexturesSet 2618
DXDraw 2619
DXDrawIndexed 2620
DXPrimiveTopologySet 2621
DXBufferSet 2622
DXShaderSet 2623
DXHandleType 2624
DXRelease 2625

33 MetaTrader per Python 2626

initialize 2632
login 2634
shutdown 2637
version 2638
last_error 2640
account_info 2642
terminal_info 2645
symbols_total 2648
symbols_get 2649
symbol_info 2652
symbol_info_tick 2656
symbol_select 2658
market_book_add 2662
market_book_get 2663
market_book_release 2666
copy_rates_from 2667
copy_rates_from_pos 2671
copy_rates_range 2674
copy_ticks_from 2677
copy_ticks_range 2680

orders_total 2683
orders_get 2684
order_calc_margin 2687
order_calc_profit 2690
order_check 2693
order_send 2697
positions_total 2702
positions_get 2703
history_orders_total 2706
history_orders_get 2708
history_deals_total 2711
history_deals_get 2713
34 Modelli ONNX 2717
Supporto ONNX 2718
Conversione Di Formato 2720
Conversione automatica del tipo di dati 2721
Creazione di un Modello 2725
Esecuzione di un modello 2733
Eseguire nello Strategy Tester 2739
OnnxCreate 2744
OnnxCreateFromBuffer 2745
OnnxRelease 2746
OnnxRun 2747
OnnxGetInputCount 2750
OnnxGetOutputCount 2751
OnnxGetInputName 2752
OnnxGetOutputName 2753
OnnxGetInputTypeInfo 2754
OnnxGetOutputTypeInfo 2755
OnnxSetInputShape 2756
OnnxSetOutputShape 2757
Strutture dati 2758
35 Libreria Standard 2761
Matematiche 2762
Statistiche 2763
Caratteristiche statistiche 2766
MathMean 2767
MathVariance 2768
MathSkewness 2769
MathKurtosis 2770
MathMoments 2771
MathMedian 2772
MathStandardDeviation 2773
MathAverageDeviation 2774
Distribuzione Normale 2775
MathProbabilityDensityNormal 2779
MathCumulativeDistributionNormal 2781
MathQuantileNormal 2783
MathRandomNormal 2785
MathMomentsNormal 2786
Distribuzione Log-normale 2787
MathProbabilityDensityLognormal 2791
MathCumulativeDistributionLognormal 2793
MathQuantileLognormal 2795
MathRandomLognormal 2797
MathMomentsLognormal 2798
La distribuzione beta 2799
MathProbabilityDensityBeta 2803

MathCumulativeDistributionBeta ... 2805
MathQuantileBeta ... 2807
MathRandomBeta ... 2809
MathMomentsBeta ... 2810
Distribuzione beta non centrale ... 2811
MathProbabilityDensityNoncentralBeta ... 2815
MathCumulativeDistributionNoncentralBeta ... 2817
MathQuantileNoncentralBeta ... 2819
MathRandomNoncentralBeta ... 2821
MathMomentsNoncentralBeta ... 2822
Distribuzione gamma ... 2823
MathProbabilityDensityGamma ... 2827
MathCumulativeDistributionGamma ... 2829
MathQuantileGamma ... 2831
MathRandomGamma ... 2833
MathMomentsGamma ... 2834
Distribuzione del Chi-quadrato ... 2835
MathProbabilityDensityChiSquare ... 2839
MathCumulativeDistributionChiSquare ... 2841
MathQuantileChiSquare ... 2843
MathRandomChiSquare ... 2845
MathMomentsChiSquare ... 2846
Distribuzione chi-quadrato non centrale ... 2847
MathProbabilityDensityNoncentralChiSquare ... 2851
MathCumulativeDistributionNoncentralChiSquare ... 2853
MathQuantileNoncentralChiSquare ... 2855
MathRandomNoncentralChiSquare ... 2857
MathMomentsNoncentralChiSquare ... 2858
Distribuzione Esponenziale ... 2859
MathProbabilityDensityExponential ... 2863
MathCumulativeDistributionExponential ... 2865
MathQuantileExponential ... 2867
MathRandomExponential ... 2869
MathMomentsExponential ... 2870
Distribuzione-F ... 2871
MathProbabilityDensityF ... 2875
MathCumulativeDistributionF ... 2877
MathQuantileF ... 2879
MathRandomF ... 2881
MathMomentsF ... 2882
Distribuzione-F noncentrale ... 2883
MathProbabilityDensityNoncentralF ... 2887
MathCumulativeDistributionNoncentralF ... 2889
MathQuantileNoncentralF ... 2891
MathRandomNoncentralF ... 2893
MathMomentsNoncentralF ... 2894
Distribuzione-T ... 2895
MathProbabilityDensityT ... 2899
MathCumulativeDistributionT ... 2901
MathQuantileT ... 2903
MathRandomT ... 2905
MathMomentsT ... 2906
Distribuzione-T non centrale ... 2907
MathProbabilityDensityNoncentralT ... 2911
MathCumulativeDistributionNoncentralT ... 2913
MathQuantileNoncentralT ... 2915
MathRandomNoncentralT ... 2917
MathMomentsNoncentralT ... 2918

Distribuzione Logistica ... 2919
MathProbabilityDensityLogistic ... 2923
MathCumulativeDistributionLogistic ... 2925
MathQuantileLogistic ... 2927
MathRandomLogistic ... 2929
MathMomentsLogistic ... 2930
Distribuzione di Cauchy ... 2931
MathProbabilityDensityCauchy ... 2935
MathCumulativeDistributionCauchy ... 2937
MathQuantileCauchy ... 2939
MathRandomCauchy ... 2941
MathMomentsCauchy ... 2942
Distribuzione uniforme ... 2943
MathProbabilityDensityUniform ... 2947
MathCumulativeDistributionUniform ... 2949
MathQuantileUniform ... 2951
MathRandomUniform ... 2953
MathMomentsUniform ... 2954
Distribuzione di Weibull ... 2955
MathProbabilityDensityWeibull ... 2959
MathCumulativeDistributionWeibull ... 2961
MathQuantileWeibull ... 2963
MathRandomWeibull ... 2965
MathMomentsWeibull ... 2966
Distribuzione binomiale ... 2967
MathProbabilityDensityBinomial ... 2970
MathCumulativeDistributionBinomial ... 2972
MathQuantileBinomial ... 2974
MathRandomBinomial ... 2976
MathMomentsBinomial ... 2977
Distribuzione binomiale negativa ... 2978
MathProbabilityDensityNegativeBinomial ... 2981
MathCumulativeDistributionNegativeBinomial ... 2983
MathQuantileNegativeBinomial ... 2985
MathRandomNegativeBinomial ... 2987
MathMomentsNegativeBinomial ... 2988
Distribuzione geometrica ... 2989
MathProbabilityDensityGeometric ... 2993
MathCumulativeDistributionGeometric ... 2995
MathQuantileGeometric ... 2997
MathRandomGeometric ... 2999
MathMomentsGeometric ... 3000
Distribuzione ipergeometrica ... 3001
MathProbabilityDensityHypergeometric ... 3005
MathCumulativeDistributionHypergeometric ... 3007
MathQuantileHypergeometric ... 3009
MathRandomHypergeometric ... 3011
MathMomentsHypergeometric ... 3012
Distribuzione di Poisson ... 3013
MathProbabilityDensityPoisson ... 3017
MathCumulativeDistributionPoisson ... 3019
MathQuantilePoisson ... 3021
MathRandomPoisson ... 3023
MathMomentsPoisson ... 3024
Subfunzioni ... 3025
MathRandomNonZero ... 3030
MathMoments ... 3031
MathPowInt ... 3032

MathFactorial ... 3033
MathTrunc ... 3034
MathRound ... 3035
MathArctan2 ... 3037
MathGamma ... 3039
MathGammaLog ... 3040
MathBeta ... 3041
MathBetaLog ... 3042
MathBetaIncomplete ... 3043
MathGammaIncomplete ... 3044
MathBinomialCoefficient ... 3045
MathBinomialCoefficientLog ... 3046
MathHypergeometric2F2 ... 3047
MathSequence ... 3048
MathSequenceByCount ... 3049
MathReplicate ... 3050
MathReverse ... 3051
MathIdentical ... 3052
MathUnique ... 3053
MathQuickSortAscending ... 3054
MathQuickSortDescending ... 3055
MathQuickSort ... 3056
MathOrder ... 3057
MathBitwiseNot ... 3058
MathBitwiseAnd ... 3059
MathBitwiseOr ... 3060
MathBitwiseXor ... 3061
MathBitwiseShiftL ... 3062
MathBitwiseShiftR ... 3063
MathCumulativeSum ... 3064
MathCumulativeProduct ... 3065
MathCumulativeMin ... 3066
MathCumulativeMax ... 3067
MathSin ... 3068
MathCos ... 3069
MathTan ... 3070
MathArcsin ... 3071
MathArccos ... 3072
MathArctan ... 3073
MathSinPi ... 3074
MathCosPi ... 3075
MathTanPi ... 3076
MathAbs ... 3077
MathCeil ... 3078
MathFloor ... 3079
MathSqrt ... 3080
MathExp ... 3081
MathPow ... 3082
MathLog ... 3083
MathLog2 ... 3084
MathLog10 ... 3085
MathLog1p ... 3086
MathDifference ... 3087
MathSample ... 3089
MathTukeySummary ... 3092
MathRange ... 3093
MathMin ... 3094
MathMax ... 3095

MathSum ... 3096
MathProduct ... 3097
MathStandardDeviation ... 3098
MathAverageDeviation ... 3099
MathMedian ... 3100
MathMean ... 3101
MathVariance ... 3102
MathSkewness ... 3103
MathKurtosis ... 3104
MathExpm1 ... 3105
MathSinh ... 3106
MathCosh ... 3107
MathTanh ... 3108
MathArcsinh ... 3109
MathArccosh ... 3110
MathArctanh ... 3111
MathSignif ... 3112
MathRank ... 3114
MathCorrelationPearson ... 3115
MathCorrelationSpearman ... 3116
MathCorrelationKendall ... 3117
MathQuantile ... 3118
MathProbabilityDensityEmpirical ... 3119
MathCumulativeDistributionEmpirical ... 3120
Logica fuzzy ... 3121
Membership functions ... 3122
CConstantMembershipFunction ... 3124
GetValue ... 3126
CCompositeMembershipFunction ... 3127
CompositionType ... 3129
MembershipFunctions ... 3129
GetValue ... 3129
CDifferencTwoSigmoidalMembershipFunction ... 3131
A1 ... 3133
A2 ... 3133
C1 ... 3134
C2 ... 3134
GetValue ... 3135
CGeneralizedBellShapedMembershipFunction ... 3136
A ... 3138
B ... 3138
C ... 3139
GetValue ... 3139
CNormalCombinationMembershipFunction ... 3140
B1 ... 3142
B2 ... 3142
Sigma1 ... 3143
Sigma2 ... 3143
GetValue ... 3144
CNormalMembershipFunction ... 3145
B ... 3147
Sigma ... 3147
GetValue ... 3148
CP_ShapedMembershipFunction ... 3149
A ... 3151
B ... 3151
C ... 3152
D ... 3152

GetValue ... 3152
CProductTwoSigmoidalMembershipFunctions ... 3154
A1 ... 3156
A2 ... 3156
C1 ... 3157
C2 ... 3157
GetValue ... 3158
CS_ShapedMembershipFunction ... 3159
A ... 3161
B ... 3161
GetValue ... 3162
CSigmoidalMembershipFunction ... 3163
A ... 3165
C ... 3165
GetValue ... 3166
CTrapezoidMembershipFunction ... 3167
X1 ... 3169
X2 ... 3169
X3 ... 3170
X4 ... 3170
GetValue ... 3171
CTriangularMembershipFunction ... 3172
X1 ... 3174
X2 ... 3174
X3 ... 3175
ToNormalMF ... 3175
GetValue ... 3175
CZ_ShapedMembershipFunction ... 3177
A ... 3179
B ... 3179
GetValue ... 3180
IMembershipFunction ... 3181
GetValue ... 3181
Fuzzy systems rules ... 3182
CMamdaniFuzzyRule ... 3183
Conclusion ... 3184
Weight ... 3184
CSugenoFuzzyRule ... 3185
Conclusion ... 3186
CSingleCondition ... 3187
Not ... 3187
Term ... 3188
Var ... 3188
CConditions ... 3190
ConditionsList ... 3190
Not ... 3191
Op ... 3191
CGenericFuzzyRule ... 3192
Conclusion ... 3192
Condition ... 3193
CreateCondition ... 3193
Fuzzy systems variables ... 3195
CFuzzyVariable ... 3196
AddTerm ... 3197
GetTermByName ... 3197
Max ... 3197
Min ... 3198
Terms ... 3198

Values ... 3199
CSugenoVariable ... 3200
Functions ... 3200
GetFuncByName ... 3201
Values ... 3201
Fuzzy terms ... 3202
MembershipFunction ... 3203
Fuzzy systems ... 3204
Mamdani system ... 3205
AggregationMethod ... 3205
Calculate ... 3206
DefuzzificationMethod ... 3206
EmptyRule ... 3206
ImplicationMethod ... 3206
Output ... 3207
OutputByName ... 3207
ParseRule ... 3207
Rules ... 3207
Sugeno system ... 3209
Calculate ... 3209
CreateSugenoFunction ... 3210
EmptyRule ... 3211
Output ... 3211
OutputByName ... 3211
ParseRule ... 3211
Rules ... 3212
OpenCL ... 3213
BufferCreate ... 3215
BufferFree ... 3216
BufferFromArray ... 3217
BufferRead ... 3218
BufferWrite ... 3219
Execute ... 3220
GetContext ... 3221
GetKernel ... 3222
GetKernelName ... 3223
GetProgram ... 3224
Initialize ... 3225
KernelCreate ... 3226
KernelFree ... 3227
SetArgument ... 3228
SetArgumentBuffer ... 3229
SetArgumentLocalMemory ... 3230
SetBuffersCount ... 3231
SetKernelsCount ... 3232
Shutdown ... 3233
SupportDouble ... 3234
Basic Class CObject ... 3235
Prev ... 3236
Prev ... 3237
Next ... 3238
Next ... 3239
Compare ... 3240
Save ... 3242
Load ... 3244
Type ... 3246
Collezione Dati ... 3247
CArray ... 3248

Step ... 3250
Step ... 3251
Total ... 3252
Available ... 3253
Max ... 3254
IsSorted ... 3255
SortMode ... 3256
Clear ... 3257
Sort ... 3258
Save ... 3259
Load ... 3260
CArrayChar ... 3261
Reserve ... 3264
Resize ... 3265
Shutdown ... 3266
Add ... 3267
AddArray ... 3268
AddArray ... 3269
Insert ... 3271
InsertArray ... 3272
InsertArray ... 3273
AssignArray ... 3275
AssignArray ... 3276
Update ... 3278
Shift ... 3279
Delete ... 3280
DeleteRange ... 3281
At ... 3282
CompareArray ... 3284
CompareArray ... 3285
InsertSort ... 3286
Search ... 3287
SearchGreat ... 3288
SearchLess ... 3289
SearchGreatOrEqual ... 3290
SearchLessOrEqual ... 3291
SearchFirst ... 3292
SearchLast ... 3293
SearchLinear ... 3294
Save ... 3295
Load ... 3296
Type ... 3298
CArrayShort ... 3299
Reserve ... 3302
Resize ... 3303
Shutdown ... 3304
Add ... 3305
AddArray ... 3306
AddArray ... 3307
Insert ... 3309
InsertArray ... 3310
InsertArray ... 3311
AssignArray ... 3313
AssignArray ... 3314
Update ... 3316
Shift ... 3317
Delete ... 3318
DeleteRange ... 3319

At ... 3320
CompareArray ... 3322
CompareArray ... 3323
InsertSort ... 3324
Search ... 3325
SearchGreat ... 3326
SearchLess ... 3327
SearchGreatOrEqual ... 3328
SearchLessOrEqual ... 3329
SearchFirst ... 3330
SearchLast ... 3331
SearchLinear ... 3332
Save ... 3333
Load ... 3335
Type ... 3337
CArrayInt ... 3338
Reserve ... 3341
Resize ... 3342
Shutdown ... 3343
Add ... 3344
AddArray ... 3345
AddArray ... 3346
Insert ... 3348
InsertArray ... 3349
InsertArray ... 3350
AssignArray ... 3352
AssignArray ... 3353
Update ... 3355
Shift ... 3356
Delete ... 3357
DeleteRange ... 3358
At ... 3359
CompareArray ... 3361
CompareArray ... 3362
InsertSort ... 3363
Search ... 3364
SearchGreat ... 3365
SearchLess ... 3366
SearchGreatOrEqual ... 3367
SearchLessOrEqual ... 3368
SearchFirst ... 3369
SearchLast ... 3370
SearchLinear ... 3371
Save ... 3372
Load ... 3374
Type ... 3376
CArrayLong ... 3377
Reserve ... 3380
Resize ... 3381
Shutdown ... 3382
Add ... 3383
AddArray ... 3384
AddArray ... 3385
Insert ... 3387
InsertArray ... 3388
InsertArray ... 3389
AssignArray ... 3391
AssignArray ... 3392

Update ... 3394
Shift ... 3395
Delete ... 3396
DeleteRange ... 3397
At ... 3398
CompareArray ... 3400
CompareArray ... 3401
InsertSort ... 3402
Search ... 3403
SearchGreat ... 3404
SearchLess ... 3405
SearchGreatOrEqual ... 3406
SearchLessOrEqual ... 3407
SearchFirst ... 3408
SearchLast ... 3409
SearchLinear ... 3410
Save ... 3411
Load ... 3413
Type ... 3415
CArrayFloat ... 3416
Delta ... 3419
Reserve ... 3420
Resize ... 3421
Shutdown ... 3422
Add ... 3423
AddArray ... 3424
AddArray ... 3425
Insert ... 3427
InsertArray ... 3428
InsertArray ... 3429
AssignArray ... 3431
AssignArray ... 3432
Update ... 3434
Shift ... 3435
Delete ... 3436
DeleteRange ... 3437
At ... 3438
CompareArray ... 3440
CompareArray ... 3441
InsertSort ... 3442
Search ... 3443
SearchGreat ... 3444
SearchLess ... 3445
SearchGreatOrEqual ... 3446
SearchLessOrEqual ... 3447
SearchFirst ... 3448
SearchLast ... 3449
SearchLinear ... 3450
Save ... 3451
Load ... 3453
Type ... 3455
CArrayDouble ... 3456
Delta ... 3459
Reserve ... 3460
Resize ... 3461
Shutdown ... 3462
Add ... 3463
AddArray ... 3464

AddArray ... 3465
Insert ... 3467
InsertArray ... 3468
InsertArray ... 3469
AssignArray ... 3471
AssignArray ... 3472
Update ... 3474
Shift ... 3475
Delete ... 3476
DeleteRange ... 3477
At ... 3478
CompareArray ... 3480
CompareArray ... 3481
Minimum ... 3482
Maximum ... 3483
InsertSort ... 3484
Search ... 3485
SearchGreat ... 3486
SearchLess ... 3487
SearchGreatOrEqual ... 3488
SearchLessOrEqual ... 3489
SearchFirst ... 3490
SearchLast ... 3491
SearchLinear ... 3492
Save ... 3493
Load ... 3495
Type ... 3497
CArrayString ... 3498
Reserve ... 3501
Resize ... 3502
Shutdown ... 3503
Add ... 3504
AddArray ... 3505
AddArray ... 3506
Insert ... 3508
InsertArray ... 3509
InsertArray ... 3510
AssignArray ... 3512
AssignArray ... 3513
Update ... 3515
Shift ... 3516
Delete ... 3517
DeleteRange ... 3518
At ... 3519
CompareArray ... 3521
CompareArray ... 3522
InsertSort ... 3523
Search ... 3524
SearchGreat ... 3525
SearchLess ... 3526
SearchGreatOrEqual ... 3527
SearchLessOrEqual ... 3528
SearchFirst ... 3529
SearchLast ... 3530
SearchLinear ... 3531
Save ... 3532
Load ... 3534
Type ... 3536

CArrayObj ... 3537
FreeMode ... 3542
FreeMode ... 3543
Reserve ... 3545
Resize ... 3546
Clear ... 3547
Shutdown ... 3548
CreateElement ... 3549
Add ... 3551
AddArray ... 3552
Insert ... 3555
InsertArray ... 3557
AssignArray ... 3559
Update ... 3561
Shift ... 3563
Detach ... 3564
Delete ... 3566
DeleteRange ... 3567
At ... 3568
CompareArray ... 3569
InsertSort ... 3570
Search ... 3571
SearchGreat ... 3572
SearchLess ... 3573
SearchGreatOrEqual ... 3574
SearchLessOrEqual ... 3575
SearchFirst ... 3576
SearchLast ... 3577
Save ... 3578
Load ... 3579
Type ... 3581
CList ... 3582
FreeMode ... 3584
FreeMode ... 3585
Total ... 3587
IsSorted ... 3588
SortMode ... 3589
CreateElement ... 3590
Add ... 3591
Insert ... 3592
DetachCurrent ... 3594
DeleteCurrent ... 3595
Delete ... 3596
Clear ... 3597
IndexOf ... 3598
GetNodeAtIndex ... 3599
GetFirstNode ... 3600
GetPrevNode ... 3601
GetCurrentNode ... 3602
GetNextNode ... 3603
GetLastNode ... 3604
Sort ... 3605
MoveToIndex ... 3606
Exchange ... 3607
CompareList ... 3608
Search ... 3609
Save ... 3610
Load ... 3612

Type ... 3614
CTreeNode ... 3615
Owner ... 3620
Left ... 3621
Right ... 3622
Balance ... 3623
BalanceL ... 3624
BalanceR ... 3625
CreateSample ... 3626
RefreshBalance ... 3627
GetNext ... 3628
SaveNode ... 3629
LoadNode ... 3630
Type ... 3631
CTree ... 3632
Root ... 3638
CreateElement ... 3639
Insert ... 3640
Detach ... 3641
Delete ... 3642
Clear ... 3643
Find ... 3644
Save ... 3645
Load ... 3646
Type ... 3647
Collezioni Dati Generali ... 3648
ICollection<T> ... 3651
Add ... 3652
Count ... 3653
Contains ... 3654
CopyTo ... 3655
Clear ... 3656
Remove ... 3657
IEqualityComparable<T> ... 3658
Equals ... 3659
HashCode ... 3660
IComparable<T> ... 3661
Compare ... 3662
IComparer<T> ... 3663
Compare ... 3664
IEqualityComparer<T> ... 3665
Equals ... 3666
HashCode ... 3667
IList<T> ... 3668
TryGetValue ... 3669
TrySetValue ... 3670
Insert ... 3671
IndexOf ... 3672
LastIndexOf ... 3673
RemoveAt ... 3674
IMap<TKey,TValue> ... 3675
Add ... 3676
Contains ... 3677
Remove ... 3678
TryGetValue ... 3679
TrySetValue ... 3680
CopyTo ... 3681
ISet<T> ... 3682

ExceptWith 3684
IntersectWith 3685
SymmetricExceptWith 3686
UnionWith 3687
IsProperSubsetOf 3688
IsProperSupersetOf 3689
IsSubsetOf 3690
IsSupersetOf 3691
Overlaps 3692
SetEquals 3693
CDefaultComparer<T> 3694
Compare 3695
CDefaultEqualityComparer<T> 3696
Equals 3697
HashCode 3698
CRedBlackTreeNode<T> 3699
Value 3700
Parent 3701
Left 3702
Right 3703
Color 3704
IsLeaf 3705
CreateEmptyNode 3706
CLinkedListNode<T> 3707
List 3708
Next 3709
Previous 3710
Value 3711
CKeyValuePair<TKey,TValue> 3712
Key 3713
Value 3714
Clone 3715
Compare 3716
Equals 3717
HashCode 3718
CArrayList<T> 3719
Capacity 3721
Count 3722
Contains 3723
TrimExcess 3724
TryGetValue 3725
TrySetValue 3726
Add 3727
AddRange 3728
Insert 3729
InsertRange 3730
CopyTo 3731
BinarySearch 3732
IndexOf 3733
LastIndexOf 3734
Clear 3735
Remove 3736
RemoveAt 3737
RemoveRange 3738
Reverse 3739
Sort 3740
CHashMap<TKey,TValue> 3741
Add 3743

Count ... 3744
Comparer ... 3745
Contains ... 3746
ContainsKey ... 3747
ContainsValue ... 3748
CopyTo ... 3749
Clear ... 3750
Remove ... 3751
TryGetValue ... 3752
TrySetValue ... 3753
CHashSet<T> ... 3754
Add ... 3756
Count ... 3757
Contains ... 3758
Comparer ... 3759
TrimExcess ... 3760
CopyTo ... 3761
Clear ... 3762
Remove ... 3763
ExceptWith ... 3764
IntersectWith ... 3765
SymmetricExceptWith ... 3766
UnionWith ... 3767
IsProperSubsetOf ... 3768
IsProperSupersetOf ... 3769
IsSubsetOf ... 3770
IsSupersetOf ... 3771
Overlaps ... 3772
SetEquals ... 3773
CLinkedList<T> ... 3774
Add ... 3776
AddAfter ... 3777
AddBefore ... 3778
AddFirst ... 3779
AddLast ... 3780
Count ... 3781
Head ... 3782
First ... 3783
Last ... 3784
Contains ... 3785
CopyTo ... 3786
Clear ... 3787
Remove ... 3788
RemoveFirst ... 3789
RemoveLast ... 3790
Find ... 3791
FindLast ... 3792
CQueue<T> ... 3793
Add ... 3794
Enqueue ... 3795
Count ... 3796
Contains ... 3797
TrimExcess ... 3798
CopyTo ... 3799
Clear ... 3800
Remove ... 3801
Dequeue ... 3802
Peek ... 3803

CRedBlackTree<T> 3804
Add 3806
Count 3807
Root 3808
Contains 3809
Comparer 3810
TryGetMin 3811
TryGetMax 3812
CopyTo 3813
Clear 3814
Remove 3815
RemoveMin 3816
RemoveMax 3817
Find 3818
FindMin 3819
FindMax 3820
CSortedMap<TKey, TValue> 3821
Add 3823
Count 3824
Comparer 3825
Contains 3826
ContainsKey 3827
ContainsValue 3828
CopyTo 3829
Clear 3830
Remove 3831
TryGetValue 3832
TrySetValue 3833
CSortedSet<T> 3834
Add 3836
Count 3837
Contains 3838
Comparer 3839
TryGetMin 3840
TryGetMax 3841
CopyTo 3842
Clear 3843
Remove 3844
ExceptWith 3845
IntersectWith 3846
SymmetricExceptWith 3847
UnionWith 3848
IsProperSubsetOf 3849
IsProperSupersetOf 3850
IsSubsetOf 3851
IsSupersetOf 3852
Overlaps 3853
SetEquals 3854
GetViewBetween 3855
GetReverse 3856
CStack<T> 3857
Add 3858
Count 3859
Contains 3860
TrimExcess 3861
CopyTo 3862
Clear 3863
Remove 3864

Push ... 3865
Peek ... 3866
Pop ... 3867
ArrayBinarySearch<T> ... 3868
ArrayIndexOf<T> ... 3869
ArrayLastIndexOf<T> ... 3870
ArrayReverse<T> ... 3871
Compare ... 3872
Equals<T> ... 3875
GetHashCode ... 3876
File ... 3879
CFile ... 3880
Handle ... 3882
Filename ... 3883
Flags ... 3884
SetUnicode ... 3885
SetCommon ... 3886
Open ... 3887
Close ... 3888
Delete ... 3889
IsExist ... 3890
Copy ... 3891
Move ... 3892
Size ... 3893
Tell ... 3894
Seek ... 3895
Flush ... 3896
IsEnding ... 3897
IsLineEnding ... 3898
FolderCreate ... 3899
FolderDelete ... 3900
FolderClean ... 3901
FileFindFirst ... 3902
FileFindNext ... 3903
FileFindClose ... 3904
CFileBin ... 3905
Open ... 3907
WriteChar ... 3908
WriteShort ... 3909
WriteInteger ... 3910
WriteLong ... 3911
WriteFloat ... 3912
WriteDouble ... 3913
WriteString ... 3914
WriteCharArray ... 3915
WriteShortArray ... 3916
WriteIntegerArray ... 3917
WriteLongArray ... 3918
WriteFloatArray ... 3919
WriteDoubleArray ... 3920
WriteObject ... 3921
ReadChar ... 3922
ReadShort ... 3923
ReadInteger ... 3924
ReadLong ... 3925
ReadFloat ... 3926
ReadDouble ... 3927
ReadString ... 3928

ReadCharArray ... 3929
ReadShortArray ... 3930
ReadIntegerArray ... 3931
ReadLongArray ... 3932
ReadFloatArray ... 3933
ReadDoubleArray ... 3934
ReadObject ... 3935
CFileTxt ... 3936
Open ... 3937
WriteString ... 3938
ReadString ... 3939
Stringhe ... 3940
CString ... 3941
Str ... 3943
Len ... 3944
Copy ... 3945
Fill ... 3946
Assign ... 3947
Append ... 3948
Insert ... 3949
Compare ... 3950
CompareNoCase ... 3951
Left ... 3952
Right ... 3953
Mid ... 3954
Trim ... 3955
TrimLeft ... 3956
TrimRight ... 3957
Clear ... 3958
ToUpper ... 3959
ToLower ... 3960
Reverse ... 3961
Find ... 3962
FindRev ... 3963
Remove ... 3964
Replace ... 3965
Oggetti Grafici ... 3966
CChartObject ... 3967
ChartId ... 3970
Window ... 3971
Name ... 3972
NumPoints ... 3973
Attach ... 3974
SetPoint ... 3975
Delete ... 3976
Detach ... 3977
ShiftObject ... 3978
ShiftPoint ... 3979
Time ... 3980
Price ... 3982
Color ... 3984
Style ... 3985
Width ... 3986
Background ... 3987
Selected ... 3988
Selectable ... 3989
Description ... 3990
Tooltip ... 3991

Timeframes ... 3992
Z_Order ... 3993
CreateTime ... 3994
LevelsCount ... 3995
LevelColor ... 3996
LevelStyle ... 3998
LevelWidth ... 4000
LevelValue ... 4002
LevelDescription ... 4004
GetInteger ... 4006
SetInteger ... 4008
GetDouble ... 4010
SetDouble ... 4012
GetString ... 4014
SetString ... 4016
Save ... 4018
Load ... 4019
Type ... 4020
Objects Lines ... 4021
CChartObjectVLine ... 4022
Create ... 4023
Type ... 4024
CChartObjectHLine ... 4025
Create ... 4026
Type ... 4027
CChartObjectTrend ... 4028
Create ... 4030
RayLeft ... 4031
RayRight ... 4032
Save ... 4033
Load ... 4034
Type ... 4035
CChartObjectTrendByAngle ... 4036
Create ... 4038
Angle ... 4039
Type ... 4040
CChartObjectCycles ... 4041
Create ... 4042
Type ... 4043
Objects Channels ... 4044
CChartObjectChannel ... 4045
Create ... 4047
Type ... 4048
CChartObjectRegression ... 4049
Create ... 4051
Type ... 4052
CChartObjectStdDevChannel ... 4053
Create ... 4055
Deviations ... 4056
Save ... 4057
Load ... 4058
Type ... 4059
CChartObjectPitchfork ... 4060
Create ... 4062
Type ... 4063
Gann Tools ... 4064
CChartObjectGannLine ... 4065
Create ... 4067

PipsPerBar 4068
Save 4069
Load 4070
Type 4071
CChartObjectGannFan 4072
Create 4074
PipsPerBar 4075
Downtrend 4076
Save 4077
Load 4078
Type 4079
CChartObjectGannGrid 4080
Create 4082
PipsPerBar 4083
Downtrend 4084
Save 4085
Load 4086
Type 4087
Fibonacci Tools 4088
CChartObjectFibo 4089
Create 4091
Type 4092
CChartObjectFiboTimes 4093
Create 4094
Type 4095
CChartObjectFiboFan 4096
Create 4097
Type 4098
CChartObjectFiboArc 4099
Create 4101
Scale 4102
Ellipse 4103
Save 4104
Load 4105
Type 4106
CChartObjectFiboChannel 4107
Create 4109
Type 4110
CChartObjectFiboExpansion 4111
Create 4113
Type 4114
Elliott Tools 4115
CChartObjectElliottWave3 4116
Create 4118
Degree 4119
Lines 4120
Save 4121
Load 4122
Type 4123
CChartObjectElliottWave5 4124
Create 4126
Type 4128
Objects Shapes 4129
CChartObjectRectangle 4130
Create 4131
Type 4132
CChartObjectTriangle 4133
Create 4134

Type ... 4135
CChartObjectEllipse ... 4136
Create ... 4137
Type ... 4138
Objects Arrows ... 4139
CChartObjectArrow ... 4140
Create ... 4142
ArrowCode ... 4144
Anchor ... 4146
Save ... 4148
Load ... 4149
Type ... 4150
Arrows with fixed code ... 4151
Create ... 4153
ArrowCode ... 4155
Type ... 4156
Objects Controls ... 4157
CChartObjectText ... 4158
Create ... 4160
Angle ... 4161
Font ... 4162
FontSize ... 4163
Anchor ... 4164
Save ... 4165
Load ... 4166
Type ... 4167
CChartObjectLabel ... 4168
Create ... 4170
X_Distance ... 4171
Y_Distance ... 4172
X_Size ... 4173
Y_Size ... 4174
Corner ... 4175
Time ... 4176
Price ... 4177
Save ... 4178
Load ... 4179
Type ... 4180
CChartObjectEdit ... 4181
Create ... 4183
TextAlign ... 4184
X_Size ... 4185
Y_Size ... 4186
BackColor ... 4187
BorderColor ... 4188
ReadOnly ... 4189
Angle ... 4190
Save ... 4191
Load ... 4192
Type ... 4193
CChartObjectButton ... 4194
State ... 4196
Save ... 4197
Load ... 4198
Type ... 4199
CChartObjectSubChart ... 4200
Create ... 4202
X_Distance ... 4203

Y_Distance ... 4204
Corner ... 4205
X_Size ... 4206
Y_Size ... 4207
Symbol ... 4208
Period ... 4209
Scale ... 4210
DateScale ... 4211
PriceScale ... 4212
Time ... 4213
Price ... 4214
Save ... 4215
Load ... 4216
Type ... 4217
CChartObjectBitmap ... 4218
Create ... 4220
BmpFile ... 4221
X_Offset ... 4222
Y_Offset ... 4223
Save ... 4224
Load ... 4225
Type ... 4226
CChartObjectBmpLabel ... 4227
Create ... 4229
X_Distance ... 4230
Y_Distance ... 4231
X_Offset ... 4232
Y_Offset ... 4233
Corner ... 4234
X_Size ... 4235
Y_Size ... 4236
BmpFileOn ... 4237
BmpFileOff ... 4238
State ... 4239
Time ... 4240
Price ... 4241
Save ... 4242
Load ... 4243
Type ... 4244
CChartObjectRectLabel ... 4245
Create ... 4247
X_Size ... 4248
Y_Size ... 4249
BackColor ... 4250
Angle ... 4251
BorderType ... 4252
Save ... 4253
Load ... 4254
Type ... 4255
Grafica Personalizzata ... 4256
CCanvas ... 4257
Attach ... 4260
Arc ... 4261
Pie ... 4265
FillPolygon ... 4269
FillEllipse ... 4270
GetDefaultColor ... 4271
ChartObjectName ... 4272

Circle ... 4273
CircleAA ... 4274
CircleWu ... 4275
Create ... 4276
CreateBitmap ... 4277
CreateBitmapLabel ... 4279
Destroy ... 4281
Ellipse ... 4282
EllipseAA ... 4283
EllipseWu ... 4284
Erase ... 4285
Fill ... 4286
FillCircle ... 4287
FillRectangle ... 4288
FillTriangle ... 4289
FontAngleGet ... 4290
FontAngleSet ... 4291
FontFlagsGet ... 4292
FontFlagsSet ... 4293
FontGet ... 4294
FontNameGet ... 4295
FontNameSet ... 4296
FontSet ... 4297
FontSizeGet ... 4298
FontSizeSet ... 4299
Height ... 4300
Line ... 4301
LineAA ... 4302
LineWu ... 4303
LineHorizontal ... 4304
LineVertical ... 4305
LineStyleSet ... 4306
LineThick ... 4307
LineThickVertical ... 4309
LineThickHorizontal ... 4310
LoadFromFile ... 4311
PixelGet ... 4312
PixelSet ... 4313
PixelSetAA ... 4314
Polygon ... 4315
PolygonAA ... 4316
PolygonWu ... 4317
PolygonThick ... 4318
PolygonSmooth ... 4319
Polyline ... 4320
PolylineSmooth ... 4321
PolylineThick ... 4322
PolylineWu ... 4323
PolylineAA ... 4324
Rectangle ... 4325
Resize ... 4326
ResourceName ... 4327
TextHeight ... 4328
TextOut ... 4329
TextSize ... 4330
TextWidth ... 4331
TransparentLevelSet ... 4332
Triangle ... 4333

TriangleAA ... 4334
TriangleWu ... 4335
Update ... 4336
Width ... 4337
CChartCanvas ... 4338
ColorBackground ... 4342
ColorBorder ... 4343
ColorText ... 4344
ColorGrid ... 4345
MaxData ... 4346
MaxDescrLen ... 4347
ShowFlags ... 4348
IsShowLegend ... 4349
IsShowScaleLeft ... 4350
IsShowScaleRight ... 4351
IsShowScaleTop ... 4352
IsShowScaleBottom ... 4353
IsShowGrid ... 4354
IsShowDescriptors ... 4355
IsShowPercent ... 4356
VScaleMin ... 4357
VScaleMax ... 4358
NumGrid ... 4359
DataOffset ... 4360
DataTotal ... 4361
DrawDescriptors ... 4362
DrawData ... 4363
Create ... 4364
AllowedShowFlags ... 4365
ShowLegend ... 4366
ShowScaleLeft ... 4367
ShowScaleRight ... 4368
ShowScaleTop ... 4369
ShowScaleBottom ... 4370
ShowGrid ... 4371
ShowDescriptors ... 4372
ShowValue ... 4373
ShowPercent ... 4374
LegendAlignment ... 4375
Accumulative ... 4376
VScaleParams ... 4377
DescriptorUpdate ... 4378
ColorUpdate ... 4379
ValuesCheck ... 4380
Redraw ... 4381
DrawBackground ... 4382
DrawLegend ... 4383
DrawLegendVertical ... 4384
DrawLegendHorizontal ... 4385
CalcScales ... 4386
DrawScales ... 4387
DrawScaleLeft ... 4388
DrawScaleRight ... 4389
DrawScaleTop ... 4390
DrawScaleBottom ... 4391
DrawGrid ... 4392
DrawChart ... 4393
CHistogramChart ... 4394

Gradient ... 4399
BarGap ... 4400
BarMinSize ... 4401
BarBorder ... 4402
Create ... 4403
SeriesAdd ... 4404
SeriesInsert ... 4405
SeriesUpdate ... 4406
SeriesDelete ... 4407
ValueUpdate ... 4408
DrawData ... 4409
DrawBar ... 4410
GradientBrush ... 4411
CLineChart ... 4412
Filled ... 4417
Create ... 4418
SeriesAdd ... 4419
SeriesInsert ... 4420
SeriesUpdate ... 4421
SeriesDelete ... 4422
ValueUpdate ... 4423
DrawChart ... 4424
DrawData ... 4425
CalcArea ... 4426
CPieChart ... 4427
Create ... 4431
SeriesSet ... 4432
ValueAdd ... 4433
ValueInsert ... 4434
ValueUpdate ... 4435
ValueDelete ... 4436
DrawChart ... 4437
DrawPie ... 4438
LabelMake ... 4439
3D graphics ... 4440
CCanvas3D ... 4441
AmbientColorGet ... 4443
AmbientColorSet ... 4444
Attach ... 4445
Create ... 4446
Destroy ... 4447
DXContext ... 4448
DXDispatcher ... 4449
InputScene ... 4450
LightColorGet ... 4451
LightColorSet ... 4452
LightDirectionGet ... 4453
LightDirectionSet ... 4454
ObjectAdd ... 4455
ProjectionMatrixGet ... 4456
ProjectionMatrixSet ... 4457
Render ... 4458
RenderBegin ... 4459
RenderEnd ... 4460
ViewMatrixGet ... 4461
ViewMatrixSet ... 4462
ViewPositionSet ... 4463
ViewRotationSet ... 4464

ViewTargetSet ... 4465
ViewUpDirectionSet ... 4466
Grafici Prezzi ... 4467
ChartID ... 4472
Mode ... 4473
Foreground ... 4474
Shift ... 4475
ShiftSize ... 4476
AutoScroll ... 4477
Scale ... 4478
ScaleFix ... 4479
ScaleFix_11 ... 4480
FixedMax ... 4481
FixedMin ... 4482
PointsPerBar ... 4483
ScalePPB ... 4484
ShowOHLC ... 4485
ShowLineBid ... 4486
ShowLineAsk ... 4487
ShowLastLine ... 4488
ShowPeriodSep ... 4489
ShowGrid ... 4490
ShowVolumes ... 4491
ShowObjectDescr ... 4492
ShowDateScale ... 4493
ShowPriceScale ... 4494
ColorBackground ... 4495
ColorForeground ... 4496
ColorGrid ... 4497
ColorBarUp ... 4498
ColorBarDown ... 4499
ColorCandleBull ... 4500
ColorCandleBear ... 4501
ColorChartLine ... 4502
ColorVolumes ... 4503
ColorLineBid ... 4504
ColorLineAsk ... 4505
ColorLineLast ... 4506
ColorStopLevels ... 4507
VisibleBars ... 4508
WindowsTotal ... 4509
WindowIsVisible ... 4510
WindowHandle ... 4511
FirstVisibleBar ... 4512
WidthInBars ... 4513
WidthInPixels ... 4514
HeightInPixels ... 4515
PriceMin ... 4516
PriceMax ... 4517
Attach ... 4518
FirstChart ... 4519
NextChart ... 4520
Open ... 4521
Detach ... 4522
Close ... 4523
BringToTop ... 4524
EventObjectCreate ... 4525
EventObjectDelete ... 4526

IndicatorAdd 4527
IndicatorDelete 4528
IndicatorsTotal 4529
IndicatorName 4530
Navigate 4531
Symbol 4532
Period 4533
Redraw 4534
GetInteger 4535
SetInteger 4536
GetDouble 4537
SetDouble 4538
GetString 4539
SetString 4540
SetSymbolPeriod 4541
ApplyTemplate 4542
ScreenShot 4543
WindowOnDropped 4544
PriceOnDropped 4545
TimeOnDropped 4546
XOnDropped 4547
YOnDropped 4548
Save 4549
Load 4550
Type 4551
Grafici Scientifici 4552
GraphPlot 4553
CAxis 4557
AutoScale 4559
Min 4560
Max 4561
Step 4562
Name 4563
Color 4564
ValuesSize 4565
ValuesWidth 4566
ValuesFormat 4567
ValuesDateTimeMode 4568
ValuesFunctionFormat 4569
ValuesFunctionFormatCBData 4571
NameSize 4572
ZeroLever 4573
DefaultStep 4574
MaxLabels 4575
MinGrace 4576
MaxGrace 4577
SelectAxisScale 4578
CColorGenerator 4579
Next 4580
Reset 4581
CCurve 4582
Type 4584
Name 4585
Color 4586
XMax 4587
XMin 4588
YMax 4589
YMin 4590

Size ... 4591
PointsSize ... 4592
PointsFill ... 4593
PointsColor ... 4594
GetX ... 4595
GetY ... 4596
LinesStyle ... 4597
LinesIsSmooth ... 4598
LinesSmoothTension ... 4599
LinesSmoothStep ... 4600
LinesEndStyle ... 4601
LinesWidth ... 4602
HistogramWidth ... 4604
CustomPlotCBData ... 4605
CustomPlotFunction ... 4606
PointsType ... 4610
StepsDimension ... 4611
TrendLineCoefficients ... 4612
TrendLineColor ... 4613
TrendLineVisible ... 4614
Update ... 4616
Visible ... 4618
CGraphic ... 4619
Create ... 4622
Destroy ... 4623
Update ... 4624
ChartObjectName ... 4625
ResourceName ... 4626
XAxis ... 4627
YAxis ... 4628
GapSize ... 4629
BackgroundColor ... 4630
BackgroundMain ... 4631
BackgroundMainSize ... 4632
BackgroundMainColor ... 4633
BackgroundSub ... 4634
BackgroundSubSize ... 4635
BackgroundSubColor ... 4636
GridLineColor ... 4637
GridBackgroundColor ... 4638
GridCircleRadius ... 4639
GridCircleColor ... 4640
GridHasCircle ... 4641
GridAxisLineColor ... 4642
HistoryNameWidth ... 4643
HistoryNameSize ... 4644
HistorySymbolSize ... 4645
TextAdd ... 4646
LineAdd ... 4647
CurveAdd ... 4648
CurvePlot ... 4651
CurvePlotAll ... 4652
CurveGetByIndex ... 4653
CurveGetByName ... 4654
CurveRemoveByIndex ... 4655
CurveRemoveByName ... 4656
CurvesTotal ... 4657
MarksToAxisAdd ... 4658

MajorMarkSize 4659
FontSet 4660
FontGet 4661
Attach 4662
CalculateMaxMinValues 4663
Height 4664
IndentDown 4665
IndentLeft 4666
IndentRight 4667
IndentUp 4668
Redraw 4669
ResetParameters 4670
ScaleX 4671
ScaleY 4672
SetDefaultParameters 4673
Width 4674
Indicatori 4675
Classi Base 4676
CSpreadBuffer 4677
Size 4679
SetSymbolPeriod 4680
At 4681
Refresh 4682
RefreshCurrent 4683
CTimeBuffer 4684
Size 4686
SetSymbolPeriod 4687
At 4688
Refresh 4689
RefreshCurrent 4690
CTickVolumeBuffer 4691
Size 4693
SetSymbolPeriod 4694
At 4695
Refresh 4696
RefreshCurrent 4697
CRealVolumeBuffer 4698
Size 4700
SetSymbolPeriod 4701
At 4702
Refresh 4703
RefreshCurrent 4704
CDoubleBuffer 4705
Size 4707
SetSymbolPeriod 4708
At 4709
Refresh 4710
RefreshCurrent 4711
COpenBuffer 4712
Refresh 4713
RefreshCurrent 4714
CHighBuffer 4715
Refresh 4716
RefreshCurrent 4717
CLowBuffer 4718
Refresh 4719
RefreshCurrent 4720
CCloseBuffer 4721

Refresh ... 4722
RefreshCurrent ... 4723
CIndicatorBuffer ... 4724
Offset ... 4726
Name ... 4727
At ... 4728
Refresh ... 4729
RefreshCurrent ... 4730
CSeries ... 4731
Name ... 4733
BuffersTotal ... 4734
Timeframe ... 4735
Symbol ... 4736
Period ... 4737
RefreshCurrent ... 4738
BufferSize ... 4739
BufferResize ... 4740
Refresh ... 4741
PeriodDescription ... 4742
CPriceSeries ... 4743
BufferResize ... 4745
GetData ... 4746
Refresh ... 4747
MinIndex ... 4748
MinValue ... 4749
MaxIndex ... 4750
MaxValue ... 4751
CIndicator ... 4752
Handle ... 4754
Status ... 4755
FullRelease ... 4756
Create ... 4757
BufferResize ... 4758
GetData ... 4759
Refresh ... 4762
Minimum ... 4763
MinValue ... 4764
Maximum ... 4765
MaxValue ... 4766
MethodDescription ... 4767
PriceDescription ... 4768
VolumeDescription ... 4769
AddToChart ... 4770
DeleteFromChart ... 4771
BarsCalculated ... 4772
CIndicators ... 4773
Create ... 4774
Refresh ... 4775
Classi Timeseries ... 4776
CiSpread ... 4777
Create ... 4779
BufferResize ... 4780
GetData ... 4781
Refresh ... 4783
CiTime ... 4784
Create ... 4786
BufferResize ... 4787
GetData ... 4788

Refresh ... 4790
CiTickVolume ... 4791
Create ... 4793
BufferResize ... 4794
GetData ... 4795
Refresh ... 4797
CiRealVolume ... 4798
Create ... 4800
BufferResize ... 4801
GetData ... 4802
Refresh ... 4804
CiOpen ... 4805
Create ... 4807
GetData ... 4808
CiHigh ... 4810
Create ... 4812
GetData ... 4813
CiLow ... 4815
Create ... 4817
GetData ... 4818
CiClose ... 4820
Create ... 4822
GetData ... 4823
Indicatori di Trend ... 4825
CiADX ... 4826
MaPeriod ... 4828
Create ... 4829
Main ... 4830
Plus ... 4831
Minus ... 4832
Type ... 4833
CiADXWilder ... 4834
MaPeriod ... 4836
Create ... 4837
Main ... 4838
Plus ... 4839
Minus ... 4840
Type ... 4841
CiBands ... 4842
MaPeriod ... 4844
MaShift ... 4845
Deviation ... 4846
Applied ... 4847
Create ... 4848
Base ... 4849
Upper ... 4850
Lower ... 4851
Type ... 4852
CiEnvelopes ... 4853
MaPeriod ... 4855
MaShift ... 4856
MaMethod ... 4857
Deviation ... 4858
Applied ... 4859
Create ... 4860
Upper ... 4861
Lower ... 4862
Type ... 4863

CiIchimoku ... 4864
TenkanSenPeriod ... 4866
KijunSenPeriod ... 4867
SenkouSpanBPeriod ... 4868
Create ... 4869
TenkanSen ... 4870
KijunSen ... 4871
SenkouSpanA ... 4872
SenkouSpanB ... 4873
ChinkouSpan ... 4874
Type ... 4875
CiMA ... 4876
MaPeriod ... 4878
MaShift ... 4879
MaMethod ... 4880
Applied ... 4881
Create ... 4882
Main ... 4883
Type ... 4884
CiSAR ... 4885
SarStep ... 4887
Maximum ... 4888
Create ... 4889
Main ... 4890
Type ... 4891
CiStdDev ... 4892
MaPeriod ... 4894
MaShift ... 4895
MaMethod ... 4896
Applied ... 4897
Create ... 4898
Main ... 4899
Type ... 4900
CiDEMA ... 4901
MaPeriod ... 4903
IndShift ... 4904
Applied ... 4905
Create ... 4906
Main ... 4907
Type ... 4908
CiTEMA ... 4909
MaPeriod ... 4911
IndShift ... 4912
Applied ... 4913
Create ... 4914
Main ... 4915
Type ... 4916
CiFrAMA ... 4917
MaPeriod ... 4919
IndShift ... 4920
Applied ... 4921
Create ... 4922
Main ... 4923
Type ... 4924
CiAMA ... 4925
MaPeriod ... 4927
FastEmaPeriod ... 4928
SlowEmaPeriod ... 4929

IndShift 4930
Applied 4931
Create 4932
Main 4933
Type 4934
CiVIDyA 4935
CmoPeriod 4937
EmaPeriod 4938
IndShift 4939
Applied 4940
Create 4941
Main 4942
Type 4943
Oscillatori 4944
CiATR 4945
MaPeriod 4947
Create 4948
Main 4949
Type 4950
CiBearsPower 4951
MaPeriod 4953
Create 4954
Main 4955
Type 4956
CiBullsPower 4957
MaPeriod 4959
Create 4960
Main 4961
Type 4962
CiCCI 4963
MaPeriod 4965
Applied 4966
Create 4967
Main 4968
Type 4969
CiChaikin 4970
FastMaPeriod 4972
SlowMaPeriod 4973
MaMethod 4974
Applied 4975
Create 4976
Main 4977
Type 4978
CiDeMarker 4979
MaPeriod 4981
Create 4982
Main 4983
Type 4984
CiForce 4985
MaPeriod 4987
MaMethod 4988
Applied 4989
Create 4990
Main 4991
Type 4992
CiMACD 4993
FastEmaPeriod 4995
SlowEmaPeriod 4996

SignalPeriod ... 4997
Applied ... 4998
Create ... 4999
Main ... 5000
Signal ... 5001
Type ... 5002
CiMomentum ... 5003
MaPeriod ... 5005
Applied ... 5006
Create ... 5007
Main ... 5008
Type ... 5009
CiOsMA ... 5010
FastEmaPeriod ... 5012
SlowEmaPeriod ... 5013
SignalPeriod ... 5014
Applied ... 5015
Create ... 5016
Main ... 5017
Type ... 5018
CiRSI ... 5019
MaPeriod ... 5021
Applied ... 5022
Create ... 5023
Main ... 5024
Type ... 5025
CiRVI ... 5026
MaPeriod ... 5028
Create ... 5029
Main ... 5030
Signal ... 5031
Type ... 5032
CiStochastic ... 5033
Kperiod ... 5035
Dperiod ... 5036
Slowing ... 5037
MaMethod ... 5038
PriceField ... 5039
Create ... 5040
Main ... 5041
Signal ... 5042
Type ... 5043
CiTriX ... 5044
MaPeriod ... 5046
Applied ... 5047
Create ... 5048
Main ... 5049
Type ... 5050
CiWPR ... 5051
CalcPeriod ... 5053
Create ... 5054
Main ... 5055
Type ... 5056
Volume Indicators ... 5057
CiAD ... 5058
Applied ... 5060
Create ... 5061
Main ... 5062

Type ... 5063
CiMFI ... 5064
MaPeriod ... 5066
Applied ... 5067
Create ... 5068
Main ... 5069
Type ... 5070
CiOBV ... 5071
Applied ... 5073
Create ... 5074
Main ... 5075
Type ... 5076
CiVolumes ... 5077
Applied ... 5079
Create ... 5080
Main ... 5081
Type ... 5082
Indicatore Bill Williams ... 5083
CiAC ... 5084
Create ... 5086
Main ... 5087
Type ... 5088
CiAlligator ... 5089
JawPeriod ... 5091
JawShift ... 5092
TeethPeriod ... 5093
TeethShift ... 5094
LipsPeriod ... 5095
LipsShift ... 5096
MaMethod ... 5097
Applied ... 5098
Create ... 5099
Jaw ... 5100
Teeth ... 5101
Lips ... 5102
Type ... 5103
CiAO ... 5104
Create ... 5106
Main ... 5107
Type ... 5108
CiFractals ... 5109
Create ... 5111
Upper ... 5112
Lower ... 5113
Type ... 5114
CiGator ... 5115
JawPeriod ... 5117
JawShift ... 5118
TeethPeriod ... 5119
TeethShift ... 5120
LipsPeriod ... 5121
LipsShift ... 5122
MaMethod ... 5123
Applied ... 5124
Create ... 5125
Upper ... 5126
Lower ... 5127
Type ... 5128

CiBWMFI ... 5129
Applied ... 5131
Create ... 5132
Main ... 5133
Type ... 5134
Indicatori Personalizzati ... 5135
NumBuffers ... 5136
NumParams ... 5137
ParamType ... 5138
ParamLong ... 5139
ParamDouble ... 5140
ParamString ... 5141
Type ... 5142
Classi di Trade ... 5143
CAccountInfo ... 5144
Login ... 5146
TradeMode ... 5147
TradeModeDescription ... 5148
Leverage ... 5149
StopoutMode ... 5150
StopoutModeDescription ... 5151
MarginMode ... 5152
MarginModeDescription ... 5153
TradeAllowed ... 5154
TradeExpert ... 5155
LimitOrders ... 5156
Balance ... 5157
Credit ... 5158
Profit ... 5159
Equity ... 5160
Margin ... 5161
FreeMargin ... 5162
MarginLevel ... 5163
MarginCall ... 5164
MarginStopOut ... 5165
Name ... 5166
Server ... 5167
Currency ... 5168
Company ... 5169
InfoInteger ... 5170
InfoDouble ... 5171
InfoString ... 5172
OrderProfitCheck ... 5173
MarginCheck ... 5174
FreeMarginCheck ... 5175
MaxLotCheck ... 5176
CSymbolInfo ... 5177
Refresh ... 5181
RefreshRates ... 5182
Name ... 5183
Select ... 5184
IsSynchronized ... 5185
Volume ... 5186
VolumeHigh ... 5187
VolumeLow ... 5188
Time ... 5189
Spread ... 5190
SpreadFloat ... 5191

TicksBookDepth ... 5192
StopsLevel ... 5193
FreezeLevel ... 5194
Bid ... 5195
BidHigh ... 5196
BidLow ... 5197
Ask ... 5198
AskHigh ... 5199
AskLow ... 5200
Last ... 5201
LastHigh ... 5202
LastLow ... 5203
TradeCalcMode ... 5204
TradeCalcModeDescription ... 5205
TradeMode ... 5206
TradeModeDescription ... 5207
TradeExecution ... 5208
TradeExecutionDescription ... 5209
SwapMode ... 5210
SwapModeDescription ... 5211
SwapRollover3days ... 5212
SwapRollover3daysDescription ... 5213
MarginInitial ... 5214
MarginMaintenance ... 5215
MarginLong ... 5216
MarginShort ... 5217
MarginLimit ... 5218
MarginStop ... 5219
MarginStopLimit ... 5220
TradeTimeFlags ... 5221
TradeFillFlags ... 5222
Digits ... 5223
Point ... 5224
TickValue ... 5225
TickValueProfit ... 5226
TickValueLoss ... 5227
TickSize ... 5228
ContractSize ... 5229
LotsMin ... 5230
LotsMax ... 5231
LotsStep ... 5232
LotsLimit ... 5233
SwapLong ... 5234
SwapShort ... 5235
CurrencyBase ... 5236
CurrencyProfit ... 5237
CurrencyMargin ... 5238
Bank ... 5239
Description ... 5240
Path ... 5241
SessionDeals ... 5242
SessionBuyOrders ... 5243
SessionSellOrders ... 5244
SessionTurnover ... 5245
SessionInterest ... 5246
SessionBuyOrdersVolume ... 5247
SessionSellOrdersVolume ... 5248
SessionOpen ... 5249

SessionClose 5250
SessionAW 5251
SessionPriceSettlement 5252
SessionPriceLimitMin 5253
SessionPriceLimitMax 5254
InfoInteger 5255
InfoDouble 5256
InfoString 5257
NormalizePrice 5258
COrderInfo 5259
Ticket 5261
TimeSetup 5262
TimeSetupMsc 5263
OrderType 5264
TypeDescription 5265
State 5266
StateDescription 5267
TimeExpiration 5268
TimeDone 5269
TimeDoneMsc 5270
TypeFilling 5271
TypeFillingDescription 5272
TypeTime 5273
TypeTimeDescription 5274
Magic 5275
PositionId 5276
VolumeInitial 5277
VolumeCurrent 5278
PriceOpen 5279
StopLoss 5280
TakeProfit 5281
PriceCurrent 5282
PriceStopLimit 5283
Symbol 5284
Comment 5285
InfoInteger 5286
InfoDouble 5287
InfoString 5288
StoreState 5289
CheckState 5290
Select 5291
SelectByIndex 5292
CHistoryOrderInfo 5293
TimeSetup 5295
TimeSetupMsc 5296
OrderType 5297
TypeDescription 5298
State 5299
StateDescription 5300
TimeExpiration 5301
TimeDone 5302
TimeDoneMsc 5303
TypeFilling 5304
TypeFillingDescription 5305
TypeTime 5306
TypeTimeDescription 5307
Magic 5308
PositionId 5309

VolumeInitial 5310
VolumeCurrent 5311
PriceOpen 5312
StopLoss 5313
TakeProfit 5314
PriceCurrent 5315
PriceStopLimit 5316
Symbol 5317
Comment 5318
InfoInteger 5319
InfoDouble 5320
InfoString 5321
Ticket 5322
SelectByIndex 5323
CPositionInfo 5324
Time 5326
TimeMsc 5327
TimeUpdate 5328
TimeUpdateMsc 5329
PositionType 5330
TypeDescription 5331
Magic 5332
Identifier 5333
Volume 5334
PriceOpen 5335
StopLoss 5336
TakeProfit 5337
PriceCurrent 5338
Commission 5339
Swap 5340
Profit 5341
Symbol 5342
Comment 5343
InfoInteger 5344
InfoDouble 5345
InfoString 5346
Select 5347
SelectByIndex 5348
SelectByMagic 5349
SelectByTicket 5350
StoreState 5351
CheckState 5352
CDealInfo 5353
Order 5355
Time 5356
TimeMsc 5357
DealType 5358
TypeDescription 5359
Entry 5360
EntryDescription 5361
Magic 5362
PositionId 5363
Volume 5364
Price 5365
Commision 5366
Swap 5367
Profit 5368
Symbol 5369

Comment ... 5370
InfoInteger ... 5371
InfoDouble ... 5372
InfoString ... 5373
Ticket ... 5374
SelectByIndex ... 5375
CTrade ... 5376
LogLevel ... 5380
SetExpertMagicNumber ... 5381
SetDeviationInPoints ... 5382
SetTypeFilling ... 5383
SetTypeFillingBySymbol ... 5384
SetAsyncMode ... 5385
SetMarginMode ... 5386
OrderOpen ... 5387
OrderModify ... 5389
OrderDelete ... 5390
PositionOpen ... 5391
PositionModify ... 5392
PositionClose ... 5393
PositionClosePartial ... 5394
PositionCloseBy ... 5396
Buy ... 5397
Sell ... 5398
BuyLimit ... 5399
BuyStop ... 5401
SellLimit ... 5402
SellStop ... 5403
Request ... 5404
RequestAction ... 5405
RequestActionDescription ... 5406
RequestMagic ... 5407
RequestOrder ... 5408
RequestSymbol ... 5409
RequestVolume ... 5410
RequestPrice ... 5411
RequestStopLimit ... 5412
RequestSL ... 5413
RequestTP ... 5414
RequestDeviation ... 5415
RequestType ... 5416
RequestTypeDescription ... 5417
RequestTypeFilling ... 5418
RequestTypeFillingDescription ... 5419
RequestTypeTime ... 5420
RequestTypeTimeDescription ... 5421
RequestExpiration ... 5422
RequestComment ... 5423
RequestPosition ... 5424
RequestPositionBy ... 5425
Result ... 5426
ResultRetcode ... 5427
ResultRetcodeDescription ... 5428
ResultDeal ... 5429
ResultOrder ... 5430
ResultVolume ... 5431
ResultPrice ... 5432
ResultBid ... 5433

ResultAsk ... 5434
ResultComment ... 5435
CheckResult ... 5436
CheckResultRetcode ... 5437
CheckResultRetcodeDescription ... 5438
CheckResultBalance ... 5439
CheckResultEquity ... 5440
CheckResultProfit ... 5441
CheckResultMargin ... 5442
CheckResultMarginFree ... 5443
CheckResultMarginLevel ... 5444
CheckResultComment ... 5445
PrintRequest ... 5446
PrintResult ... 5447
FormatRequest ... 5448
FormatRequestResult ... 5449
CTerminalInfo ... 5450
Build ... 5452
IsConnected ... 5453
IsDLLsAllowed ... 5454
IsTradeAllowed ... 5455
IsEmailEnabled ... 5456
IsFtpEnabled ... 5457
MaxBars ... 5458
CodePage ... 5459
CPUCores ... 5460
MemoryPhysical ... 5461
MemoryTotal ... 5462
MemoryAvailable ... 5463
MemoryUsed ... 5464
IsX64 ... 5465
OpenCLSupport ... 5466
DiskSpace ... 5467
Language ... 5468
Name ... 5469
Company ... 5470
Path ... 5471
DataPath ... 5472
CommonDataPath ... 5473
InfoInteger ... 5474
InfoString ... 5475
Moduli Strategia ... 5476
Classi base per Expert Advisors ... 5479
CExpertBase ... 5480
InitPhase ... 5482
TrendType ... 5483
UsedSeries ... 5484
EveryTick ... 5485
Open ... 5486
High ... 5487
Low ... 5488
Close ... 5489
Spread ... 5490
Time ... 5491
TickVolume ... 5492
RealVolume ... 5493
Init ... 5494
Symbol ... 5495

Period ... 5496
Magic ... 5497
ValidationSettings ... 5498
SetPriceSeries ... 5499
SetOtherSeries ... 5500
InitIndicators ... 5501
InitOpen ... 5502
InitHigh ... 5503
InitLow ... 5504
InitClose ... 5505
InitSpread ... 5506
InitTime ... 5507
InitTickVolume ... 5508
InitRealVolume ... 5509
PriceLevelUnit ... 5510
StartIndex ... 5511
CompareMagic ... 5512
CExpert ... 5513
Init ... 5518
Magic ... 5519
InitSignal ... 5520
InitTrailing ... 5521
InitMoney ... 5522
InitTrade ... 5523
Deinit ... 5524
OnTickProcess ... 5525
OnTradeProcess ... 5526
OnTimerProcess ... 5527
OnChartEventProcess ... 5528
OnBookEventProcess ... 5529
MaxOrders ... 5530
Signal ... 5531
ValidationSettings ... 5532
InitIndicators ... 5533
OnTick ... 5534
OnTrade ... 5535
OnTimer ... 5536
OnChartEvent ... 5537
OnBookEvent ... 5538
InitParameters ... 5539
DeinitTrade ... 5540
DeinitSignal ... 5541
DeinitTrailing ... 5542
DeinitMoney ... 5543
DeinitIndicators ... 5544
Refresh ... 5545
Processing ... 5546
SelectPosition ... 5548
CheckOpen ... 5549
CheckOpenLong ... 5550
CheckOpenShort ... 5551
OpenLong ... 5552
OpenShort ... 5553
CheckReverse ... 5554
CheckReverseLong ... 5555
CheckReverseShort ... 5556
ReverseLong ... 5557
ReverseShort ... 5559

CheckClose 5561
CheckCloseLong 5562
CheckCloseShort 5563
CloseAll 5564
Close 5565
CloseLong 5566
CloseShort 5567
CheckTrailingStop 5568
CheckTrailingStopLong 5569
CheckTrailingStopShort 5570
TrailingStopLong 5571
TrailingStopShort 5572
CheckTrailingOrderLong 5573
CheckTrailingOrderShort 5574
TrailingOrderLong 5575
TrailingOrderShort 5576
CheckDeleteOrderLong 5577
CheckDeleteOrderShort 5578
DeleteOrders 5579
DeleteOrder 5580
DeleteOrderLong 5581
DeleteOrderShort 5582
LotOpenLong 5583
LotOpenShort 5584
LotReverse 5585
PrepareHistoryDate 5586
HistoryPoint 5587
CheckTradeState 5588
WaitEvent 5589
NoWaitEvent 5590
TradeEventPositionStopTake 5591
TradeEventOrderTriggered 5592
TradeEventPositionOpened 5593
TradeEventPositionVolumeChanged 5594
TradeEventPositionModified 5595
TradeEventPositionClosed 5596
TradeEventOrderPlaced 5597
TradeEventOrderModified 5598
TradeEventOrderDeleted 5599
TradeEventNotIdentified 5600
TimeframeAdd 5601
TimeframesFlags 5602
CExpertSignal 5603
BasePrice 5606
UsedSeries 5607
Weight 5608
PatternsUsage 5609
General 5610
Ignore 5611
Invert 5612
ThresholdOpen 5613
ThresholdClose 5614
PriceLevel 5615
StopLevel 5616
TakeLevel 5617
Expiration 5618
Magic 5619
ValidationSettings 5620

InitIndicators ... 5621
AddFilter ... 5622
CheckOpenLong ... 5623
CheckOpenShort ... 5624
OpenLongParams ... 5625
OpenShortParams ... 5626
CheckCloseLong ... 5627
CheckCloseShort ... 5628
CloseLongParams ... 5629
CloseShortParams ... 5630
CheckReverseLong ... 5631
CheckReverseShort ... 5632
CheckTrailingOrderLong ... 5633
CheckTrailingOrderShort ... 5634
LongCondition ... 5635
ShortCondition ... 5636
Direction ... 5637
CExpertTrailing ... 5638
CheckTrailingStopLong ... 5640
CheckTrailingStopShort ... 5641
CExpertMoney ... 5642
Percent ... 5644
ValidationSettings ... 5645
CheckOpenLong ... 5646
CheckOpenShort ... 5647
CheckReverse ... 5648
CheckClose ... 5649
I moduli di Segnali di Trade ... 5650
Signals of the Indicator Accelerator Oscillator ... 5653
Signals of the Indicator Adaptive Moving Average ... 5656
Signals of the Indicator Awesome Oscillator ... 5660
Signals of the Oscillator Bears Power ... 5664
Signals of the Oscillator Bulls Power ... 5666
Signals of the Oscillator Commodity Channel Index ... 5668
Signals of the Oscillator DeMarker ... 5672
Signals of the Indicator Double Exponential Moving Average ... 5676
Signals of the Indicator Envelopes ... 5680
Signals of the Indicator Fractal Adaptive Moving Average ... 5683
Signals of the Intraday Time Filter ... 5687
Signals of the Oscillator MACD ... 5689
Signals of the Indicator Moving Average ... 5695
Signals of the Indicator Parabolic SAR ... 5699
Signals of the Oscillator Relative Strength Index ... 5701
Signals of the Oscillator Relative Vigor Index ... 5707
Signals of the Oscillator Stochastic ... 5709
Signals of the Oscillator Triple Exponential Average ... 5714
Signals of the Indicator Triple Exponential Moving Average ... 5718
Signals of the Oscillator Williams Percent Range ... 5722
Trailing Stop Classes ... 5725
CTrailingFixedPips ... 5726
StopLevel ... 5728
ProfitLevel ... 5729
ValidationSettings ... 5730
CheckTrailingStopLong ... 5731
CheckTrailingStopShort ... 5732
CTrailingMA ... 5733
Period ... 5735
Shift ... 5736

Method ... 5737
Applied ... 5738
InitIndicators ... 5739
ValidationSettings ... 5740
CheckTrailingStopLong ... 5741
CheckTrailingStopShort ... 5742
CTrailingNone ... 5743
CheckTrailingStopLong ... 5744
CheckTrailingStopShort ... 5745
CTrailingPSAR ... 5746
Step ... 5748
Maximum ... 5749
InitIndicators ... 5750
CheckTrailingStopLong ... 5751
CheckTrailingStopShort ... 5752
Classi Money Management ... 5753
CMoneyFixedLot ... 5754
Lots ... 5756
ValidationSettings ... 5757
CheckOpenLong ... 5758
CheckOpenShort ... 5759
CMoneyFixedMargin ... 5760
CheckOpenLong ... 5761
CheckOpenShort ... 5762
CMoneyFixedRisk ... 5763
CheckOpenLong ... 5764
CheckOpenShort ... 5765
CMoneyNone ... 5766
ValidationSettings ... 5767
CheckOpenLong ... 5768
CheckOpenShort ... 5769
CMoneySizeOptimized ... 5770
DecreaseFactor ... 5772
ValidationSettings ... 5773
CheckOpenLong ... 5774
CheckOpenShort ... 5775
Pannelli e Dialoghi ... 5776
CRect ... 5778
Left ... 5779
Top ... 5780
Right ... 5781
Bottom ... 5782
Width ... 5783
Height ... 5784
SetBound ... 5785
Move ... 5786
Shift ... 5787
Contains ... 5788
Format ... 5789
CDateTime ... 5790
MonthName ... 5792
ShortMonthName ... 5793
DayName ... 5794
ShortDayName ... 5795
DaysInMonth ... 5796
DateTime ... 5797
Date ... 5798
Time ... 5799

Sec ... 5800
Min ... 5801
Hour ... 5802
Day ... 5803
Mon ... 5804
Year ... 5805
SecDec ... 5806
SecInc ... 5807
MinDec ... 5808
MinInc ... 5809
HourDec ... 5810
HourInc ... 5811
DayDec ... 5812
DayInc ... 5813
MonDec ... 5814
MonInc ... 5815
YearDec ... 5816
YearInc ... 5817
CWnd ... 5818
Create ... 5822
Destroy ... 5823
OnEvent ... 5824
OnMouseEvent ... 5825
Name ... 5826
ControlsTotal ... 5827
Control ... 5828
ControlFind ... 5829
Rect ... 5830
Left ... 5831
Top ... 5832
Right ... 5833
Bottom ... 5834
Width ... 5835
Height ... 5836
Move ... 5837
Shift ... 5838
Resize ... 5839
Contains ... 5840
Alignment ... 5841
Align ... 5842
Id ... 5843
IsEnabled ... 5844
Enable ... 5845
Disable ... 5846
IsVisible ... 5847
Visible ... 5848
Show ... 5849
Hide ... 5850
IsActive ... 5851
Activate ... 5852
Deactivate ... 5853
StateFlags ... 5854
StateFlagsSet ... 5855
StateFlagsReset ... 5856
PropFlags ... 5857
PropFlagsSet ... 5858
PropFlagsReset ... 5859
MouseX ... 5860

MouseY ... 5861
MouseFlags ... 5862
MouseFocusKill ... 5863
OnCreate ... 5864
OnDestroy ... 5865
OnMove ... 5866
OnResize ... 5867
OnEnable ... 5868
OnDisable ... 5869
OnShow ... 5870
OnHide ... 5871
OnActivate ... 5872
OnDeactivate ... 5873
OnClick ... 5874
OnChange ... 5875
OnMouseDown ... 5876
OnMouseUp ... 5877
OnDragStart ... 5878
OnDragProcess ... 5879
OnDragEnd ... 5880
DragObjectCreate ... 5881
DragObjectDestroy ... 5882
CWndObj ... 5883
OnEvent ... 5885
Text ... 5886
Color ... 5887
ColorBackground ... 5888
ColorBorder ... 5889
Font ... 5890
FontSize ... 5891
ZOrder ... 5892
OnObjectCreate ... 5893
OnObjectChange ... 5894
OnObjectDelete ... 5895
OnObjectDrag ... 5896
OnSetText ... 5897
OnSetColor ... 5898
OnSetColorBackground ... 5899
OnSetFont ... 5900
OnSetFontSize ... 5901
OnSetZOrder ... 5902
OnDestroy ... 5903
OnChange ... 5904
CWndContainer ... 5905
Destroy ... 5907
OnEvent ... 5908
OnMouseEvent ... 5909
ControlsTotal ... 5910
Control ... 5911
ControlFind ... 5912
Add ... 5913
Delete ... 5914
Move ... 5915
Shift ... 5916
Id ... 5917
Enable ... 5918
Disable ... 5919
Show ... 5920

Hide ... 5921
MouseFocusKill ... 5922
Save ... 5923
Load ... 5924
OnResize ... 5925
OnActivate ... 5926
OnDeactivate ... 5927
CLabel ... 5928
Create ... 5933
OnSetText ... 5934
OnSetColor ... 5935
OnSetFont ... 5936
OnSetFontSize ... 5937
OnCreate ... 5938
OnShow ... 5939
OnHide ... 5940
OnMove ... 5941
CBmpButton ... 5942
Create ... 5948
Border ... 5949
BmpNames ... 5950
BmpOffName ... 5951
BmpOnName ... 5952
BmpPassiveName ... 5953
BmpActiveName ... 5954
Pressed ... 5955
Locking ... 5956
OnSetZOrder ... 5957
OnCreate ... 5958
OnShow ... 5959
OnHide ... 5960
OnMove ... 5961
OnChange ... 5962
OnActivate ... 5963
OnDeactivate ... 5964
OnMouseDown ... 5965
OnMouseUp ... 5966
CButton ... 5967
Create ... 5974
Pressed ... 5975
Locking ... 5976
OnSetText ... 5977
OnSetColor ... 5978
OnSetColorBackground ... 5979
OnSetColorBorder ... 5980
OnSetFont ... 5981
OnSetFontSize ... 5982
OnCreate ... 5983
OnShow ... 5984
OnHide ... 5985
OnMove ... 5986
OnResize ... 5987
OnMouseDown ... 5988
OnMouseUp ... 5989
CEdit ... 5990
Create ... 5995
ReadOnly ... 5996
TextAlign ... 5997

OnObjectEndEdit 5998
OnSetText 5999
OnSetColor 6000
OnSetColorBackground 6001
OnSetColorBorder 6002
OnSetFont 6003
OnSetFontSize 6004
OnSetZOrder 6005
OnCreate 6006
OnShow 6007
OnHide 6008
OnMove 6009
OnResize 6010
OnChange 6011
OnClick 6012
CPanel 6013
Create 6018
BorderType 6019
OnSetText 6020
OnSetColorBackground 6021
OnSetColorBorder 6022
OnCreate 6023
OnShow 6024
OnHide 6025
OnMove 6026
OnResize 6027
OnChange 6028
CPicture 6029
Create 6034
Border 6035
BmpName 6036
OnCreate 6037
OnShow 6038
OnHide 6039
OnMove 6040
OnChange 6041
CScroll 6042
Create 6044
OnEvent 6045
MinPos 6046
MaxPos 6047
CurrPos 6048
CreateBack 6049
CreateInc 6050
CreateDec 6051
CreateThumb 6052
OnClickInc 6053
OnClickDec 6054
OnShow 6055
OnHide 6056
OnChangePos 6057
OnThumbDragStart 6058
OnThumbDragProcess 6059
OnThumbDragEnd 6060
CalcPos 6061
CScrollV 6062
CreateInc 6068
CreateDec 6069

CreateThumb 6070
OnResize 6071
OnChangePos 6072
OnThumbDragStart 6073
OnThumbDragProcess 6074
OnThumbDragEnd 6075
CalcPos 6076
CScrollH 6077
CreateInc 6083
CreateDec 6084
CreateThumb 6085
OnResize 6086
OnChangePos 6087
OnThumbDragStart 6088
OnThumbDragProcess 6089
OnThumbDragEnd 6090
CalcPos 6091
CWndClient 6092
Create 6095
OnEvent 6096
ColorBackground 6097
ColorBorder 6098
BorderType 6099
VScrolled 6100
HScrolled 6101
CreateBack 6102
CreateScrollV 6103
CreateScrollH 6104
OnResize 6105
OnVScrollShow 6106
OnVScrollHide 6107
OnHScrollShow 6108
OnHScrollHide 6109
OnScrollLineDown 6110
OnScrollLineUp 6111
OnScrollLineLeft 6112
OnScrollLineRight 6113
Rebound 6114
CListView 6115
Create 6121
OnEvent 6122
TotalView 6123
AddItem 6124
Select 6125
SelectByText 6126
SelectByValue 6127
Value 6128
CreateRow 6129
OnResize 6130
OnVScrollShow 6131
OnVScrollHide 6132
OnScrollLineDown 6133
OnScrollLineUp 6134
OnItemClick 6135
Redraw 6136
RowState 6137
CheckView 6138
CComboBox 6139

Create ... 6145
OnEvent ... 6146
AddItem ... 6147
ListViewItems ... 6148
Select ... 6149
SelectByText ... 6150
SelectByValue ... 6151
Value ... 6152
CreateEdit ... 6153
CreateButton ... 6154
CreateList ... 6155
OnClickEdit ... 6156
OnClickButton ... 6157
OnChangeList ... 6158
ListShow ... 6159
ListHide ... 6160
CCheckBox ... 6161
Create ... 6167
OnEvent ... 6168
Text ... 6169
Color ... 6170
Checked ... 6171
Value ... 6172
CreateButton ... 6173
CreateLabel ... 6174
OnClickButton ... 6175
OnClickLabel ... 6176
CCheckGroup ... 6177
Create ... 6183
OnEvent ... 6184
AddItem ... 6185
Value ... 6186
CreateButton ... 6187
OnVScrollShow ... 6188
OnVScrollHide ... 6189
OnScrollLineDown ... 6190
OnScrollLineUp ... 6191
OnChangeItem ... 6192
Redraw ... 6193
RowState ... 6194
CRadioButton ... 6195
Create ... 6197
OnEvent ... 6198
Text ... 6199
Color ... 6200
State ... 6201
CreateButton ... 6202
CreateLabel ... 6203
OnClickButton ... 6204
OnClickLabel ... 6205
CRadioGroup ... 6206
Create ... 6212
OnEvent ... 6213
AddItem ... 6214
Value ... 6215
CreateButton ... 6216
OnVScrollShow ... 6217
OnVScrollHide ... 6218

OnScrollLineDown ... 6219
OnScrollLineUp ... 6220
OnChangeItem ... 6221
Redraw ... 6222
RowState ... 6223
Select ... 6224
CSpinEdit ... 6225
Create ... 6231
OnEvent ... 6232
MinValue ... 6233
MaxValue ... 6234
Value ... 6235
CreateEdit ... 6236
CreateInc ... 6237
CreateDec ... 6238
OnClickInc ... 6239
OnClickDec ... 6240
OnChangeValue ... 6241
CDialog ... 6242
Create ... 6244
OnEvent ... 6245
Caption ... 6246
Add ... 6247
CreateWhiteBorder ... 6248
CreateBackground ... 6249
CreateCaption ... 6250
CreateButtonClose ... 6251
CreateClientArea ... 6252
OnClickCaption ... 6253
OnClickButtonClose ... 6254
ClientAreaVisible ... 6255
ClientAreaLeft ... 6256
ClientAreaTop ... 6257
ClientAreaRight ... 6258
ClientAreaBottom ... 6259
ClientAreaWidth ... 6260
ClientAreaHeight ... 6261
OnDialogDragStart ... 6262
OnDialogDragProcess ... 6263
OnDialogDragEnd ... 6264
CAppDialog ... 6265
Create ... 6268
Destroy ... 6269
OnEvent ... 6270
Run ... 6271
ChartEvent ... 6272
Minimized ... 6273
IniFileSave ... 6274
IniFileLoad ... 6275
IniFileName ... 6276
IniFileExt ... 6277
CreateCommon ... 6278
CreateExpert ... 6279
CreateIndicator ... 6280
CreateButtonMinMax ... 6281
OnClickButtonClose ... 6282
OnClickButtonMinMax ... 6283
OnAnotherApplicationClose ... 6284

Rebound 6285
Minimize 6286
Maximize 6287
CreateInstanceId 6288
ProgramName 6289
SubwinOff 6290
36 Il passaggio dall' MQL4 6291
37 List of MQL5 Functions 6295
38 List of MQL5 Constants 6330

Riferimento MQL5

MetaQuotes Language 5 (MQL5) è un linguaggio ad alto-livello progettato per lo sviluppo di indicatori tecnici, robot di trading e applicazioni di utilità, che automatizzano il trading finanziario. MQL5 è stato sviluppato da MetaQuotes per la loro piattaforma di trading. La sintassi del linguaggio è molto vicina al C++, consentendo ai programmatori di sviluppare applicazioni nello stile di programmazione orientata agli oggetti (OOP).

Oltre al linguaggio MQL5, il pacchetto della piattaforma di trading include anche il MetaEditor IDE con strumenti di scrittura del codice estremamente avanzati, quali templates, snippet, debugging, profiling e strumenti di completamento automatico, così come MQL5 Storage che abilita il controllo delle versioni dei file.

Il supporto al linguaggio è disponibile sul sito Web MQL5.community, che contiene un numero enorme di CodeBase gratuito e una pletora di articoli. Questi articoli coprono tutti gli aspetti del trading moderno, tra cui reti neurali, statistiche e analisi, trading ad alta frequenza, arbitraggio, testing ed ottimizzazione delle strategie di trading, utilizzo di robot di automazione di trading e altro ancora.

Gli sviluppatori di programmi di trading e MQL5 possono comunicare sul forum, ordinare e sviluppare applicazioni usando il servizio Freelance, così come acquistare e vendere programmi protetti nel Market di applicazioni di trading automatico.

Il linguaggio MQL5 fornisce specializzate funzioni di trading e predefiniti gestori di eventi per aiutare i programmatori a sviluppare Expert Advisors (EA), che controllano automaticamente i processi di trading seguendo specifiche regole di trading. Oltre agli EA, MQL5 consente lo sviluppo personalizzato di indicatori tecnici, script e librerie.

Questo riferimento al linguaggio MQL5 contiene funzioni, operazioni, parole riservate ed altre costruzioni del linguaggio, suddivisi in categorie. Il riferimento fornisce anche le descrizioni delle classi della Libreria Standard utilizzate per lo sviluppo di strategie di trading, pannelli di controllo, grafica personalizzata e abilitazione dell'accesso ai file.

Inoltre, il CodeBase contiene la libreria di analisi numerica ALGLIB, che può essere utilizzata per risolvere vari problemi matematici.

Tipi di applicazioni MQL5

I programmi MQL5 sono suddivisi in cinque tipi specializzati basati sulle attività di automazione del trading che implementano:

- **Un Expert Advisor** è un sistema di trading automatico collegato ad un grafico-chart. Un Expert Advisor contiene gestori(handlers) di evento per gestire eventi predefiniti che attivano l'esecuzione di elementi di strategia di trading appropriati. Ad esempio, un evento di inizializzazione e deinizializzazione del programma, nuovi ticks, eventi timer, modifiche alla profondità di mercato(Depth of Market), eventi chart e personalizzati.
  Oltre a calcolare i segnali di trading in base alle regole implementate, gli Expert Advisors possono anche eseguire automaticamente le operazioni e inviarle direttamente ad un server di trading. Gli Expert Advisors sono memorizzati in *<Directory_del_Terminale>\MQL5\Experts*.
- **Un Indicatore personalizzato** è un indicatore tecnico sviluppato da un utente in aggiunta agli indicatori standard integrati nella piattaforma di trading. Gli indicatori personalizzati, oltre a quelli standard, non possono fare il trade automaticamente, ma implementano solo funzioni analitiche. Gli

indicatori personalizzati possono utilizzare i valori di altri indicatori per i calcoli e possono essere chiamati da Expert Advisors.
Gli indicatori personalizzati sono memorizzati in <*Directory_del_Terminale*>*MQL5**Indicators*.

- **Uno Script** è un programma per una singola esecuzione di un'azione. A differenza degli Expert Advisors, gli script non gestiscono alcun evento tranne per i trigger(inneschi), inizializzazione e deinitializzazione. Un codice di script deve contenere la funzione di gestore di OnStart. Gli script sono memorizzati in <*Directory_del_Terminale*>*MQL5**Scripts*.
- **Servizio** è un programma che, a differenza degli indicatori, degli Expert Advisors e degli script, non richiede di essere associato ad un chart per funzionare. Come gli script, i servizi non gestiscono alcun evento tranne che per l'attivazione(innesco/trigger). Per avviare un servizio, il suo codice dovrebbe contenere la funzione dell handler OnStart. I servizi non accettano altri eventi tranne Start, ma sono in grado di inviare eventi personalizzati ai charts usando EventChartCustom. I servizi sono memorizzati in <*Directory_del_Terminale*>*MQL5**Services*.
- **Una Libreria**è un insieme di funzioni personalizzate. Le librerie sono destinate a memorizzare e distribuire algoritmi comunemente usati di programmi personalizzati.
Le librerie sono memorizzate in <*Directory_del_Terminale*>*MQL5**Libraries*.
- **Un file Include** è un testo di sorgente dei blocchi più utilizzati di programmi personalizzati. Tali file possono essere inclusi nei testi dei sorgenti di Expert Advisors, script, indicatori personalizzati e librerie nella fase di compilazione. L'uso di file inclusi è più preferibile rispetto all'uso delle librerie a causa di un carico aggiuntivo che si verifica quando si chiamano le funzioni della libreria.
I file Include possono essere memorizzati nella stessa directory in cui si trova il file originale. In questo caso viene usata la direttiva #include con le virgolette. Un'altra opzione è quella di memorizzare i file include nella <*Directory_del_Terminale*>*MQL5**Include*. In questo caso #include deve essere usato con parentesi angolari.

Le basi del Linguaggio

Il Linguaggio MetaQuotes 5 (MQL5) è un linguaggio di programmazione object-oriented ad alto livello destinato alla scrittura di strategie di trading automatiche, indicatori tecnici personalizzati per l'analisi dei vari mercati finanziari. Permette non solo di scrivere una varietà di sistemi esperti, progettati per funzionare in tempo reale, ma anche creare i propri strumenti grafici per aiutare a prendere decisioni di trade.

MQL5 si basa sul concetto del popolare linguaggio di programmazione C++. Rispetto ad MQL4, il nuovo linguaggio ha ora enumerazioni, strutture, classi e event handlers. Aumentando il numero degli inclusi tipi main, l'interazione di programmi eseguibili in MQL5 con altre applicazioni attraverso le dll è ora il più semplice possibile. La sintassi MQL5 è simile alla sintassi del C++, e questo lo rende facile tradurlo in programmi informatici da moderni linguaggi di programmazione.

Per aiutarvi a studiare il linguaggio MQL5, tutti gli argomenti sono raggruppati nelle seguenti sezioni:

- Sintassi
- Tipi di Dati
- Operazioni ed Espressioni
- Operatori
- Funzioni
- Variabili
- Preprocessore
- Programmazione Ad Oggetti (OOP Object-Oriented-Programming)
- Namespaces

Sintassi

Per quanto riguarda la sintassi, il linguaggio MQL5 per la progrmamazione di strategie di trading, è molto simile al linguaggio di programmazione C++ , fatta eccezione per alcune caratteristiche:

- nessun indirizzo aritmetico;
- nessun operatore goto;
- un' enumerazione anonima non può essere dichiarata;
- non multiple ereditarietà.

Vedi anche

Enumerazioni, Strutture e Classi, Eredità

Commenti

I commenti multi-riga iniziano con la coppia di simboli /* e termina con */ questa. Tali tipi di commenti non possono essere nidificati. Commenti a riga singola iniziano con la coppia di simboli // e termina con il carattere di nuova riga, essi possono essere annidati in altri commenti su più righe. I commenti sono consentiti ovunque dove gli spazi siano consentiti, e possono avere qualsiasi numero di spazi tra loro.

Esempi:

```
//--- Commento a riga-singola
/*  Commento-
    Multi- // riga singola di commento, annidata
    riga
*/
```

Identificatori

Gli identificatori vengono usati come nomi di variabili e funzioni. La lunghezza dell'identificatore non può superare 63 caratteri.

I caratteri consentiti da scrivere in un identificatore: cifre 0-9, lettere latine maiuscole e minuscole a-z ed A-Z, riconosciute come diversi caratteri, il carattere di sottolineatura (_). Il primo carattere non può essere un numero.

L'identificatore non deve coincidere con la parolareserved.

Esempi:

```
NAME1 name1 Total_5 Paper
```

Vedi anche

Variabili, Funzioni

Parole riservate

I seguenti identificatori sono registrati come parole riservate, ciascuno di essi corrisponde ad una certa azione, e non può essere utilizzato in un altro significato:

Tipi di Dati

bool	float	uint
char	int	ulong
class	long	union
color	short	ushort
datetime	string	void
double	struct	
enum	uchar	

Specificatori di accesso

const	private	virtual
delete	protected	
override	public	

Classi di memoria

extern	input	static

Operatori

break	dynamic_cast	operator
case	else	pack
continue	for	return
default	if	sizeof
delete	new	switch
do	offsetof	while

Altro

this	#define	#import
true	#ifdef	#include

this	#define	#import
false	#ifndef	#property
template	#else	group
typename	#endif	namespace

Tipi di Dati

Qualsiasi programma opera con i dati. I dati possono essere di tipi diversi a seconda dei loro fini. Ad esempio, i dati integer(_*interi) vengono utilizzati per accedere ai componenti degli array. I dati sui prezzi appartengono a quelli di doppia precisione con virgola mobile. Ciò è dovuto al fatto che nessun particolare tipo di dati per dati sui prezzi, è fornito in MQL5.

I dati di differenti tipi sono trattati con tassi differenti. I dati integer vengono elaborati come i più veloci. Per elaborare i dati a doppia precisione, una speciale co-processore viene utilizzato. However, because of complexity of internal representation of data with floating point, they are processed slower than the integer ones.

Dati stringa vengono elaborati in modo più lungo presso a causa dell' allocazione/riallocazione dinamica della memoria del computer.

I tipi di dati di base sono:

- interi (char, short, int, long, uchar, ushort, uint, ulong);
- logici (bool);
- letterali (ushort);
- stringa (string);
- numeri in virgola-mobile (double, float);
- colori (color);
- data ed orario (datetime);
- enumerazioni (enum).

Tipi di dati complessi sono i seguenti:

- strutture;
- classi.

In termini di OOP i tipi di dati complessi sono chiamati tipi di dati astratti.

I tipi *color* e *datetime* hanno senso solo per facilitare la visualizzazione e l'input dei parametri definiti dal di fuori - dalla tabella di Expert Advisor o delle proprietà degli indicatori personalizzati (la scheda Inputs). I dati *color* e *datetime* vengono rappresentati come interi. I tipi interi ed in virgola-mobile vengono chiamati tipi aritmetici (numerici).

Vengono usati solo i type casting impliciti nelle espressioni, a meno che non viene specificato il casting esplicito.

Vedi anche

Typecasting

Tipi Integer

In MQL5 gli interi sono rappresentati da undici tipi. Alcuni tipi possono essere usati insieme ad altri, se richiesto dalla logica del programma, ma in questo caso è necessario ricordare le regole di conversione di tipo.

La tabella seguente elenca le caratteristiche di ogni tipo. Inoltre, l'ultima colonna mostra il tipo in C++ corrispondente a ciascun tipo.

Tipo	Dimensione in Byte	Valore Minimo	Valore Massimo	Analogo C++
char	1	-128	127	char
uchar	1	0	255	unsigned char, BYTE
bool	1	0(false)	1(true)	bool
short	2	-32 768	32 767	short, wchar_t
ushort	2	0	65 535	unsigned short, WORD
int	4	- 2 147 483 648	2 147 483 647	int
uint	4	0	4 294 967 295	unsigned int, DWORD
color	4	-1	16 777 215	int, COLORREF
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	__int64
ulong	8	0	18 446 744 073 709 551 615	unsigned __int64
datetime	8	0 (1970.01.01 0:00:00)	32 535 244 799 (3000.12.31 23:59:59)	__time64_t

Valori di tipo intero possono essere presentati come costanti numeriche, letterali di colore, letterali data-ora, caratteri costanti ed enumerazioni.

Vedi anche

Conversione Dati, Costanti dei Tipi Numerici

Tipi Char, Short, Int e Long

char

Il tipo *char* prende 1 byte di memoria (8 bit) e consente di esprimere in notazione binaria 2^8=256 valori. Il tipo *char* può contenere valori sia positivi che negativi. L'intervallo di valori va da -128 a 127.

uchar

Il tipo intero *uchar* occupa anche 1 byte di memoria, così come il tipo *char*, ma a differenza da esso, *uchar* è designato solo per i valori positivi. Il valore minimo è zero, il valore massimo è 255. La prima lettera u del nome del tipo *uchar* è l'abbreviazione di *unsigned*.

short

La dimensione del tipo *short* è 2 byte (16 bit) e, di conseguenza, consente di esprimere l'intervallo di valori pari a 2 alla potenza di16: 2^16=65 536. Siccome il tipo *short* è con il segno, e contiene i valori sia positivi che negativi, l'intervallo di valori è compreso tra -32 768 e 32 767.

ushort

Il tipo *short* senza segno, è il tipo *ushort*, che anche, ha una dimensione di 2 byte. Il valore minimo è 0, il valore massimo è 65 535.

int

La dimensione del tipo *int* è di 4 byte (32 bit). Il valore minimo è di -2 147 483 648, quello massimo è di 2 147 483 647.

uint

Il tipo intero senza segno è *uint*. Ci vogliono 4 byte di memoria e consente di esprimere interi da 0 a 4 294 967 295.

long

La grandezza del tipo *long* è di 8 byte (64 bit). Il valore minimo è -9 223 372 036 854 775 808, il valore massimo è di 9 223 372 036 854 775 807.

ulong

Il tipo *ulong* anche, occupa 8 byte e può memorizzare i valori da 0 a 18 446 744 073 709 551 615.

Esempi:

```
char  ch=12;
short sh=-5000;
int   in=2445777;
```

Poiché i tipi interi senza segno non sono designati per memorizzare i valori negativi, il tentativo di impostare un valore negativo può portare a conseguenze inaspettate. Un tale semplice script porterà ad un ciclo infinito:

```
//--- Loop infinito
voidOnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<128;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
     }
  }
```

La variante corretta è:

```
//--- Variante corretta
voidOnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<=127;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
      if(ch==127) break;
     }
  }
```

Risultato:

```
   ch= -128  u_ch= 128
   ch= -127  u_ch= 129
   ch= -126  u_ch= 130
   ch= -125  u_ch= 131
   ch= -124  u_ch= 132
   ch= -123  u_ch= 133
   ch= -122  u_ch= 134
   ch= -121  u_ch= 135
   ch= -120  u_ch= 136
   ch= -119  u_ch= 137
   ch= -118  u_ch= 138
   ch= -117  u_ch= 139
   ch= -116  u_ch= 140
   ch= -115  u_ch= 141
   ch= -114  u_ch= 142
   ch= -113  u_ch= 143
   ch= -112  u_ch= 144
```

```
   ch= -111  u_ch= 145
    ...
```

Esempi:

```
//--- Valori negativi non possono essere memorizzati in tipi senza segno
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

Esadecimale: numeri 0-9, le lettere a-f o A-F per i valori di 10-15, iniziano con 0x o 0X.

Esempi:

```
0x0A, 0x12, 0X12, 0x2f, 0xA3, 0Xa3, 0X7C7
```

Per le variabili intere, i valori possono essere impostati in forma binaria utilizzando il prefisso B. Ad esempio, è possibile codificare le ore di lavoro di una sessione di trading in variabile di tipo int ed utilizzare le informazioni su di esse secondo l'algoritmo richiesto:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta 1 per le ore di lavoro e 0 per quelle non lavorative
   int AsianSession   =B'111111111'; // Sessione Asiatica da 00:00 alle 9:00
   int EuropeanSession=B'111111111000000000'; // Sessione Europea 9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // Sessione Americana 16:00 - 02
//--- derive numerical values of the sessions
   PrintFormat("Ore della sessione Asiatica come valore =%d",AsianSession);
   PrintFormat("Ore della sessione Europea come valore è %d",EuropeanSession);
   PrintFormat("Ore della sessione Americana come valore è %d %d",AmericanSession);
//--- e ora visualizziamo le rappresentazioni stringa delle ore di lavoro delle sessio
   Print("Sessione Asiatica ",GetHoursForSession(AsianSession));
   Print("Sessione Europea ",GetHoursForSession(EuropeanSession));
   Print("Sessione Americana ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------------------+
//| restituisce le ore di lavoro della sessione come stringa             |
//+------------------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- al fine di verificare, usale l'orperazione bit AND e slittamenti sinistro di 1 b
//--- avviare il controllo dal bit più basso
   int bit=1;
   string out="ore lavorative: ";
//--- controlla tutti i 24 bit a partire da quello zero e fino a 23, inclusivamente
   for(int i=0;i<24;i++)
     {
```

```
      //--- riceve lo stato dei bit in numeri
      bool workinghour=(session&bit)==bit;
      //--- aggiunge il numero di ore al messaggio
      if(workinghour )out=out+StringFormat("%d ",i);
      //--- slitta di un bit a sinistra e controlla il valore di quello successivo
      bit<<=1;
     }
//--- stringa risultato
   return out;
  }
```

Vedi anche

Typecasting

Costanti Carattere

I caratteri come elementi di un stringa in MQL5 sono indici nel set di caratteri Unicode. Sono valori esadecimali che possono essere espressi in numeri interi, e che possono essere manipolati da operazioni integer come addizione e sottrazione.

Qualsiasi carattere singolo tra virgolette o un codice esadecimale ASCII di un carattere '\x10' è un carattere costante ed è di tipo ushort. Ad esempio, un record di tipo '0' è un valore numerico 30, che corrisponde all'indice di zero nella tabella di caratteri.

Esempio:

```
voidOnStart()
  {
//--- definizione delle costanti carattere
   int symbol_0='0';
   int symbol_9=symbol_0+9; // ottengo simbolo '9'
//--- valori output delle costanti
   printf("In a decimal form: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("In a hexadecimal form: symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symbol_
//--- inserire costanti in una stringa
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- questo è ciò che sembrano in una stringa
   Print(test);
  }
```

Il backslash è un carattere di controllo per il compilatore quando si tratta di stringhe costanti e le costanti di caratteri nel testo sorgente di un programma. Alcuni simboli, per esempio un singolo apice ('), i doppi apici ("), backslash (\) e caratteri di controllo possono essere rappresentati come una combinazione di simboli che iniziano con un backslash (\), secondo la seguente tabella:

Nome del carattere	Codice mnemonico o immagine	Record in MQL5	Valore numerico
nuova linea (line feed)	LF	'\n'	10
tabulazione orizzontale	HT	'\t'	9
ritorno a capo	CR	'\r'	13
backslash	\	'\\'	92
apice singolo	'	'\''	39
virgolette o apice doppio	"	'\"'	34
codice esadecimale	hhhh	'\xhhhh'	Da 1 a 4 caratteri esadecimali

Nome del carattere	Codice mnemonico o immagine	Record in MQL5	Valore numerico
codice decimale	d	'\d'	numero decimale da 0 a 65535

Se il backslash è seguito da un carattere diverso da quelli sopra descritti, il risultato risultato è indefinito.

Esempio

```
voidOnStart()
  {
//--- dichiarazione delle costanti carattere
   int a='A';
   int b='$';
   int c='©';       // codice 0xA9
   int d='\xAE';    // codice per il simbolo ®
//--- print output delle costanti
   Print(a,b,c,d);
//--- aggiungere un carattere alla stringa
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- sostituire un carattere in una stringa
   StringSetCharacter(test,0,b);
   Print(test);
//--- sostituire un carattere in una stringa
   StringSetCharacter(test,0,c);
   Print(test);
//--- sostituire un carattere in una stringa
   StringSetCharacter(test,0,d);
   Print(test);
//--- rappresentare i caratteri come un numero
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- aggiungere un carattere alla stringa
   StringSetCharacter(test,1,a1);
   Print(test);
//--- aggiungere un carattere alla stringa
   StringSetCharacter(test,1,b1);
   Print(test);
//--- aggiungere un carattere alla stringa
   StringSetCharacter(test,1,c1);
   Print(test);
//--- aggiungere un carattere alla stringa
   StringSetCharacter(test,1,d1);
```

```
   Print(test);
  }
```

Come è stato menzionato sopra, il valore di una costante (o variabile) carattere è un indice nella tabella dei caratteri. L' indice essendo un intero, esso può essere scritto in modi diversi.

```
voidOnStart()
  {
//---
   int a=0xAE;      // il codice di ® corrisponde al letterale '\xAE'
   int b=0x24;      // il codice di $ corrisponde al letterale '\x24'
   int c=0xA9;      // il codice di © corrisponde al letterale '\xA9'
   int d=0x263A;    // il codice di ☺ corrisponde al letterale '\x263A'
//--- mostra valori
   Print(a,b,c,d);
//--- aggiungere un carattere alla stringa
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- sostituire un carattere in una stringa
   StringSetCharacter(test,0,b);
   Print(test);
//--- sostituire un carattere in una stringa
   StringSetCharacter(test,0,c);
   Print(test);
//--- sostituire un carattere in una stringa
   StringSetCharacter(test,0,d);
   Print(test);
//--- codici di suits
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- aggiungere il carattere di picche
   StringSetCharacter(test,1,a1);
   Print(test);
//--- aggiungere il carattere di cuori
   StringSetCharacter(test,2,b1);
   Print(test);
//--- aggiungere il carattere di quadri
   StringSetCharacter(test,3,c1);
   Print(test);
//--- aggiungere il carattere di bastoni
   StringSetCharacter(test,4,d1);
   Print(test);
//--- esempio di letterali carattere in una stringa
   test="Regina\x2660Ace\x2662";
   printf("%s",test);
  }
```

La rappresentazione interna di un carattere letterale è il tipoushort. Le costanti carattere possono accettare valori da 0 a 65535.

Vedi anche

StringSetCharacter(), StringGetCharacter(), ShortToString(), ShortArrayToString(), StringToShortArray()

Tipo Datetime

Il tipo datetime è destinato a memorizzare la data e l'ora come il numero di secondi trascorsi dal 01 gennaio 1970. Questo tipo occupa 8 byte di memoria.

Costanti di data e ora possono essere rappresentate come una stringa letterale, che consiste in 6 parti che mostrano il valore numerico dell 'anno, mese, giorno (o giorno, mese, anno), ore, minuti e secondi. La costante è racchiusa tra virgolette singole e inizia con il carattere D.

I valori vanno dal 1 ° gennaio 1970 al 31 dicembre 3000. Sia data (anno, mese, giorno) che orario (ore, minuti, secondi), o tutti insieme possono essere omessi.

Con la specificazione della data letterale, è desiderabile che si specifichi l'anno, mese e giorno. Altrimenti il compilatore restituisce un avviso su una voce incompleta.

Esempi:

```
datetime NY=D'2015.01.01 00:00';      // Orario dell'inizio dell'anno 2015
datetime d1=D'1980.07.19 12:30:27';   // Anno Mese Giorno Ore Minuti Secondi
datetime d2=D'19.07.1980 12:30:27';   // Equivale a D'1980.07.19 12:30:27';
datetime d3=D'19.07.1980 12';         // Equivale a D'1980.07.19 12:00:00'
datetime d4=D'01.01.2004';            // Equivale aD'01.01.2004 00:00:00'
datetime compilation_date=__DATE__;              // Data di compilazione
datetime compilation_date_time=__DATETIME__;     // Data ed orario di compilazione
datetime compilation_time=__DATETIME__-__DATE__;// Orario di compilazione
//--- Examples of declarations after which compiler warnings will be returned
datetime warning1=D'12:30:27';        // Equivale a D'[data di compilazione] 12:30:27'
datetime warning2=D'';                // Equivale a __DATETIME__
```

Vedi anche

Struttura del tipo Data, Data ed Ora, TimeToString, StringToTime

Color Type

Il tipo colore è stato progettato per la memorizzazione delle informazioni sul colore ed occupa 4 byte in memoria. Il primo byte viene ignorato, i restanti 3 byte contengono i-componenti RGB.

Costanti di colore possono essere rappresentate in tre modi: letteralmente, da numeri interi, o per nome (per i colori nominati Web-colori solamente).

La rappresentazione letterale è composta da tre parti che rappresentano i valori dei tassi numerici dei tre principali componenti di colore: rosso, verde, blu. La costante inizia con C ed è racchiusa tra apici. I tassi dei valori numerici di una componente di colore giacciono nell'intervallo da 0 a 255.

Integer-valued representation is written in a form of hexadecimal or a decimal number. Un numero esadecimale sembra 0x00BBGGRR, dove RR è il tasso del componente di colore rosso, GG - del verde, e BB - di quello blu. Costanti decimali non si riflettono direttamente nel RGB. Esse rappresentano un valore decimale della rappresentazione esadecimale intera.

Colori specifici riflettono il cosiddetto Web-color impostato.

Esempi:

```
//--- Letterali
C'128,128,128'    // Grigio
C'0x00,0x00,0xFF' // Blu
//nomi dei colori
clrRed            // rosso
clrYellow         // giallo
clrBlack          // nero
//--- Rappresentazioni integrali
0xFFFFFF          // Bianco
16777215          // Bianco
0x008000          // Verde
32768             // Verde
```

Vedi anche

Web Colors,ColorToString, StringToColor, Typecasting

Tipo Bool

Il tipo bool è designato per memorizzare i valori logici true o false, la rappresentazione numerica di essi è 1 o 0, rispettivamente.

Esempi:

```
bool a = true;
bool b = false;
bool c = 1;
```

La rappresentazione interna è un numero intero largo 1 byte. Va notato che nelle espressioni logiche è possibile utilizzare altro numero intero o tipi reali o espressioni di questi tipi - il compilatore non genererà alcun errore. In questo caso, il valore zero viene interpretato come false, e tutti gli altri valori - come true.

Esempi:

```
   int i=5;
   double d=-2.5;
   if(i) Print("i = ",i," ed è impostato a true");
   else Print("i = ",i," ed è impostato a false");

   if(d) Print("d = ",d," ed ha il valore true");
   else Print("d = ",d," ed ha il valore false");

   i=0;
   if(i) Print("i = ",i," ed ha il valore true");
   else Print("i = ",i," ed ha il valore false");

   d=0.0;
   if(d) Print("d = ",d," ed ha il valore true");
   else Print("d = ",d," ed ha il valore false");

//--- Risultato dell'esecuzione
//   i= 5 ed ha il valore true
//   d= -2.5 ed ha il valore true
//   i= 0 ed ha il valore false
//   d= 0 ed ha il valore false
```

Vedi anche

Operazioni Booleane, Precedenza Regole

Enumerazioni

Dati del tipo enum appartengono ad un certo insieme limitato di dati. Definizione del tipo di enumerazione:

```
enum nome del tipo enumerabile
  {
  elenco di valori
  };
```

L'elenco dei valori è un elenco di identificatori di costanti denominate, separate da virgole.

Esempio:

```
enum mese// enumerazione di costanti denominate
   {
   Gennaio,
   Febbraio,
   Marzo,
   Aprile,
   Maggio,
   Giugno,
   Luglio,
   Agosto,
   Settembre,
   Ottobre,
   Novembre,
   Dicembre
   };
```

Dopo l'enumerazione viene dichiarata, appare un nuovo tipo di dati integer-a-valore-4-byte. La dichiarazione del nuovo tipo di dati consente al compilatore di controllare rigorosamente i tipi di parametri passati, perchè l'enumerazione introduce nuove costanti denominate. Nel precedente esempio, la costante denominata Gennaio ha il valore di 0, Febbraio - 1, Dicembre - 11.

Regola: Se un certo valore non è assegnato ad una costante nominata che è un membro dell'enumerazione, il nuovo valore viene formato automaticamente. Se è il primo membro dell'enumerazione, ad esso verrà assegnato il valore 0. Per tutti i membri successivi, i valori vengono calcolati sulla base del valore dei componenti precedenti aggiungendo uno.

Esempio:

```
enum intervals  // Enumerazione delle costanti nominate
   {
    month=1,     // Intervallo di un mese
    two_months,  // Due mesi
    quarter,     // Tre mesi - un quarto di anno
    halfyear=6,  // Mezzo anno
    year=12,     // Anno - 12 mesi
   };
```

Note

- A differenza di C++, la grandezza della rappresentazione interna del tipo enumerato in MQL5 è sempre uguale a 4 byte. Che è, sizeof (months) restituisce il valore 4.
- A differenza del C++, un' enumerazione anonima non può essere dichiarata in MQL5. Vale a dire, un nome univoco deve essere sempre specificato dopo la parola chiave enum.

Vedi anche

Typecasting

Tipi Reali (double, float)

I tipi reali (o tipi a virgola mobile) rappresentano i valori con una parte frazionaria. Nel linguaggio MQL5 ci sono due tipi di numeri floating point(_* a virgola mobile) .Il metodo di rappresentazione dei numeri reali nella memoria del computer è definito dallo standard IEEE 754 ed è indipendente dalle piattaforme, sistemi operativi o linguaggi di programmazione.

Tipo	Dimensione in byte	Valore Minimo Positivo	Valore Massimo	Analogo C++
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.2250738585072014e-308	1.7976931348623158e+308	double

double

double tipo di numero reale che occupa 64 bit (1 bit di segno, 11 bit di esponente e 52 bit di mantissa).

float

float tipo di numero reale che occupa 32 bit (1 bit di segno, 11 bit di esponente e 23 bit di mantissa).

vector

Array unidimensionale di numeri di tipo double. La memoria per i dati viene allocata dinamicamente. Le proprietà del vettore possono essere ottenute utilizzando i metodi, con la quale la dimensione del vettore può essere modificata. La voce vector<double> può essere usata nelle funzioni dei template.

vectorf

Array unidimensionale di numeri di tipo float che può essere utilizzato al posto di vector se la perdita di precisione non ha importanza. La voce vector<float> può essere usata nelle funzioni dei template.

vectorc

Array unidimensionale di numeri di tipo complex è pensato per gestire numeri complessi. La voce vector<complex> può essere usata nelle funzioni dei template. Le operazioni sui vettori di tipo vectorc non sono ancora state implementate.

matrix

Matrix è un array bidimensionale di numeri di tipo double. La memoria per gli elementi della matrice è distribuita dinamicamente. Le proprietà della matrice possono essere ottenute utilizzando i metodi,

con la quale la forma della matrice può essere cambiata. La voce matrix<double> può essere usata nelle funzioni dei template.

matrixf

Array bidimensionale di numeri di tipo float che può essere utilizzato al posto di matrix se la perdita di precisione non ha importanza. La voce matrix<float> può essere usata nelle funzioni dei template.

matrixc

Array bidimensionale di numeri di tipo complex è pensato per gestire numeri complessi. La voce matrix<complex> può essere usata nelle funzioni dei template. Le operazioni sulle matrici di tipo matrixc non sono ancora state implementate.

Il nome double significa che la precisione di questi numeri è due volte l'accuratezza del tipo di numerifloat. Nella maggior parte dei casi, il tipo double è il più conveniente. In molti casi la precisione limitata dei numeri float non è sufficiente. Il motivo per cui il tipo float è ancora usato, è per salvare la memoria (questo è importante per grandi array di numeri reali).

Costanti in floating-point consistono di una parte intera, un punto (.) e la parte frazionaria. Le parti intera e frazionaria sono sequenze di cifre decimali.

Esempi:

```
double a=12.111;
double b=-956.1007;
float  c =0.0001;
float  d =16;
```

C'è un modo scientifico di scrivere costanti reali, spesso questo metodo di registrazione è più compatto di quello tradizionale.

Esempio:

```
double c1=1.12123515e-25;
double c2=0.000000000000000000000000112123515; // 24 zero dopo il punto decimale

Print("1. c1 =",DoubleToString(c1,16));
// Result: 1. c1 = 0.0000000000000000

Print("2. c1 =",DoubleToString(c1,-16));
// Result: 2. c1 = 1.1212351499999999e-025

Print("3. c2 =",DoubleToString(c2,-16));
// Result: 3. c2 = 1.1212351499999999e-025
```

Va ricordato che i numeri reali sono memorizzati nella memoria con una certa precisione limitata nel sistema binario, mentre generalmente viene utilizzata la notazione decimale . Ecco perché molti numeri che sono appunto rappresentati nel sistema decimale possono essere scritti solo come una frazione infinita nel sistema binario.

Per esempio, i numeri 0,3 e 0,7 sono rappresentati nel computer come frazioni infinite, mentre il numero di 0,25 è memorizzato esattamente così, perché rappresenta la potenza di due.

A questo proposito, si raccomanda di non confrontare due numeri reali per l'uguaglianza, in quanto tale confronto non è corretto.

Esempio:

```
voidOnStart()
  {
//---
   double three=3.0;
   double x,y,z;
   x=1/three;
   y=4/three;
   z=5/three;
   if(x+y==z)
      Print("1/3 + 4/3 == 5/3");
   else
      Print("1/3 + 4/3 != 5/3");
// Risultato: 1/3 + 4/3 != 5/3
  }
```

Se si ha ancora bisogno di confrontare l'uguaglianza di due numeri reali, allora si può fare in due modi diversi. Il primo modo consiste nel confrontare la differenza tra due numeri con qualche piccola quantità che specifica la precisione del confronto.

Esempio:

```
bool EqualDoubles(double d1,double d2,double epsilon)
  {
   if(epsilon<0)
      epsilon=-epsilon;
//---
   if(d1-d2>epsilon)
      return false;
   if(d1-d2<-epsilon)
      return false;
//---
   return true;
  }
voidOnStart()
  {
   double d_val=0.7;
   float  f_val=0.7;
   if(EqualDoubles(d_val,f_val,0.000000000000001))
      Print(d_val," equals ",f_val);
   else
      Print("Different: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToStrinq
// Risultato: Different: d_val= 0.7000000000000000   f_val= 0.6999999880790710
  }
```

Osservato che il valore di epsilon nell'esempio precedente non può essere inferiore alla costante predefinita DBL_EPSILON. Il valore di questa costante è 2.2204460492503131e-016. La corrispondente costante per il tipo float è FLT_EPSILON = 1.192092896e-07. Il significato di questi valori è il seguente: è il valore più basso che soddisfa la condizione 1.0 + DBL_EPSILON! = 1.0 (per numeri di tipo float 1.0 + FLT_EPSILON! = 1.0).

Il secondo modo offre il confronto della differenza normalizzata di due numeri reali con zero. Non ha senso confrontare la differenza di numeri normalizzati con uno zero, perché qualsiasi operazione matematica con i numeri normalizzati da un risultato non-normalizzato.

Esempio:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0)
      return(true);
   else
      return(false);
  }
voidOnStart()
  {
   double d_val=0.3;
   float  f_val=0.3;
   if(CompareDoubles(d_val,f_val))
      Print(d_val," equals ",f_val);
   else
      Print("Different: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToStrin
// Result: Different: d_val= 0.3000000000000000   f_val= 0.3000000119209290
  }
```

Alcune operazioni del co-processore matematico possono provocare il numero reale valido, che non può essere utilizzato in operazioni matematiche e operazioni di confronto, poiché il risultato di operazioni con numeri reali non validi è indefinito. Per esempio, quando si cerca di calcolare l' arcoseno di 2, il risultato è l'infinito negativo.

Esempio:

```
   double abnormal = MathArcsin(2.0);
   Print("MathArcsin(2.0) =",abnormal);
// Risultato:  MathArcsin(2.0) = -1.#IND
```

Oltre al meno infinito c'è il più infinito e NaN (not a number). Per determinare che questo numero non è valido, è possibile utilizzare MathIsValidNumber(). Secondo lo standard IEEE, hanno una particolare rappresentazione macchina. Per esempio, più infinito per il tipo double ha la rappresentanza bit di 0x7FF0 0000 0000 0000.

Esempi:

```
struct str1
  {
   double d;
  };
```

```
struct str2
  {
   long l;
  };

//--- Start
   str1 s1;
   str2 s2;
//---
   s1.d=MathArcsin(2.0);         // Ottiene il numero non valido -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;      // numero non valido -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;      // il più grande non-numero SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;      // il più piccolo non-numero QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;      // il più grande non-numero QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;      // Infinito positivo 1.#INF ed il più piccolo non nume
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;      // Infinito negativo -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;      // Zero negativo -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;      // 0.5
   s1=s2;
   printf("9.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FF0000000000000;      // 1.0
   s1=s2;
   printf("10.  %f %I64X",s1.d,s2.l);
//---
```

```
   s2.l=0x7FEFFFFFFFFFFFFF;     // Il più grande numero normalizzato (MAX_DBL)
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
//---
   s2.l=0x0010000000000000;     // Il più piccolo positivo normalizzato (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
   s1.d=0.7;                    // Mostra il numero di 0.7 - frazione senza fine
   s2=s1;
   printf("13.  %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNAN0 FFFF000000000000
3.   1.#SNAN0 7FF7000000000000
4.   1.#QNAN0 7FF8000000000000
5.   1.#QNAN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FEFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000
13.  6.9999999999999996e-001 3FE6666666666666
*/
```

Vedi anche

DoubleToString, NormalizeDouble, Costanti dei Tipi Numerici

Numero complesso (complex)

Il tipo complex incorporato è una struttura con due campi double:

```
struct complex
  {
   double            real;    // Real part
   double            imag;    // Imaginary part
  };
```

Il tipo "complex" può essere passato per valore come parametro per le funzioni MQL5 (a differenza delle strutture ordinarie, che vengono passate solo per riferimento). Per le funzioni importate da DLL, il tipo "complex" deve essere passato solo per riferimento.

Il suffisso 'i' è usato per descrivere costanti complesse:

```
complex square(complex c)
  {
   return(c*c);
  }
void OnStart()
  {
   Print(square(1+2i));  // A constant is passed as a parameter
  }
// "(-3,4)" will be output, which is a string representation of the complex number
```

Attualmente sono disponibili solo le operazioni semplici per i numeri complessi: =, +, -, *, /, +=, -=, *=, /=, ==,!=.

Il supporto per ulteriori funzioni matematiche verrà aggiunto in seguito, consentendo il calcolo del valore assoluto, seno, coseno e altro.

vectorc

Array unidimensionale di numeri di tipo complex è pensato per gestire numeri complessi. La voce vector<complex> può essere usata nelle funzioni dei template. Le operazioni sui vettori di tipo vectorc non sono ancora state implementate.

matrix

Matrix è un array bidimensionale di numeri di tipo double. La memoria per gli elementi della matrice è distribuita dinamicamente. Le proprietà della matrice possono essere ottenute utilizzando i metodi, con la quale la forma della matrice può essere cambiata. La voce matrix<double> può essere usata nelle funzioni dei template.

matrixf

Array bidimensionale di numeri di tipo float che può essere utilizzato al posto di matrix se la perdita di precisione non ha importanza. La voce matrix<float> può essere usata nelle funzioni dei template.

matrixc

Array bidimensionale di numeri di tipo complex è pensato per gestire numeri complessi. La voce matrix<complex> può essere usata nelle funzioni dei template. Le operazioni sulle matrici di tipo matrixc non sono ancora state implementate.

Tipo Stringa

Il tipo stringa viene utilizzato per la memorizzazione di stringhe di testo. Una stringa di testo è una sequenza di caratteri in formato Unicode con lo zero finale al termine di essa. Una costante stringa può essere assegnata ad una variabile stringa. Una costante stringa è una sequenza di caratteri Unicode racchiuse tra virgolette: "Questa è una costante stringa".

Se è necessario includere le virgolette (") in una stringa, il carattere di backslash (\) deve essere messo prima. Qualsiasi speciali costanti carattere possono essere scritte in una stringa, se il carattere di backslash (\) viene digitato prima di esse.

Esempi:

```
string svar="Questo è un carattere stringa";
string svar2=StringSubstr(svar,0,4);
Print("Simbolo del Copyright\t\x00A9");
FileWrite(handle,"Questa stringa contiene il simbolo di nuova riga \n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

Per rendere il codice sorgente leggibile, lunghe costanti stringa possono essere suddivise in parti senza operazione di addizione. Durante la compilazione, queste parti verranno combinate in una stringa long:

```
//--- Dichiarare una costante stringa long
   string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"'
                    " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
                    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
                    "<head>\n"
                    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ut
                    "<title>Report Operazioni di Trade</title>\n"
                    "</head>";
//--- Fai l'output della costante stringa nel log
   Print(HTML_head);
  }
```

Vedi anche

Funzioni di Conversione, Funzioni di Stringa,FileOpen, FileReadString, FileWriteString

Strutture, Classi e Interfacce

Strutture

Una struttura è un insieme di elementi di qualsiasi tipo (ad eccezione del tipo void). Così, la struttura combina dati logicamente correlati di tipo diverso.

Dichiarazione Struttura

Il tipo di dati struttura è determinato dalla seguente descrizione:

```
struct structure_name
  {
   elements_description
  };
```

Il nome struttura non può essere usato come un identificatore (nome di una variabile o funzione). Va notato che in MQL5 elementi strutturali si susseguono direttamente, senza allineamento. In C++ un tale ordine è fatto per il compilatore utilizzando la seguente istruzione:

```
#pragma pack(1)
```

Se si vuole fare un altro allineamento nella struttura, utilizzare i membri ausiliari, "fillers" per la giusta dimensione.

Esempio:

```
struct trade_settings
  {
   uchar  slippage;     // valore grandezza-slippage permissibile 1 byte
   char   reserved1;    // salta 1 byte
   short  reserved2;    // salta 2 bytes
   int    reserved4;    // altri 4 bytes vengono saltati. garantisce l'allineamento de
   double take;         // valore del prezzo del fissaggio del profitto
   double stop;         // valore del prezzo dello stop protettivo
  };
```

Tale descrizione delle strutture allineate è necessaria solo per il trasferimento di funzioni importate dll.

Attenzione: Questo esempio illustra dati non correttamente designati. Sarebbe meglio prima dichiarare *take* e *stop* dati di grandi dimensioni del tipo double, e quindi dichiarare lo *slippage*membro del tipo uchar. In questo caso, la rappresentazione interna dei dati sarà sempre la stessa indipendentemente dal valore specificato in *#pragma pack()*.

Se una struttura contiene variabili di tipo string e/o oggetto di un array dinamico, il compilatore assegna un costruttore implicito a tale struttura. Questo costruttore resetta tutti i membri della struttura di tipo string ed inizializza correttamente oggetti della matrice dinamica.

Strutture Semplici

Le strutture che non contengono stringhe, oggetti delle classi, puntatori ed oggetti di array dinamici sono chiamate Strutture Semplici. Le variabili di strutture semplici e le loro array possono essere passate come parametri alle funzioni importate da DLL.

La copia di strutture semplici è consentita solo in due casi:

- Se gli oggetti appartengono allo stesso tipo di struttura
- Se gli oggetti sono collegati dalla linea che significa che una struttura è un discendente di un'altra.

Per fornire un esempio, sviluppiamo la struttura personalizzata CustomMqlTick con il suo contenuto identico a quella built-in MqlTick. Il compilatore non consente di copiare il valore dell'oggetto MqlTick nell'oggetto di tipo CustomMqlTick. Typecasting diretto al tipo necessario provoca anche l'errore di compilazione:

```
      //--- la copia di strutture semplici di diversi tipi è vietata
      my_tick1=last_tick;               // il compilatore restituisce un errore qui

      //--- fare il typecasting di strutture di tipi differenti l'un l'altra è vietato
      my_tick1=(CustomMqlTick)last_tick;// il compilatore restituisce un errore qui
```

Pertanto, resta una sola opzione - la copia dei valori degli elementi della struttura uno ad uno. È comunque permesso copiare i valori dello stesso tipo di CustomMqlTick.

```
      CustomMqlTick my_tick1,my_tick2;
      //--- è consentito copiare gli oggetti dello stesso tipo di CustomMqlTick nel se
      my_tick2=my_tick1;

      //--- creare un array dagli oggetti della struttura semplice CustomMqlTick e sc
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

La funzione ArrayPrint() è chiamata per un controllo per visualizzare la funzione *arr[] valore array nel journal*.

```
//+------------------------------------------------------------------+
//| Funzione start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
// --- sviluppa la struttura simile al built-in MqlTick
   struct CustomMqlTick
     {
      datetime          time;          // Ora di aggiornamento del prezzo Last
      double            bid;           // Prezzo Bid corrente
      double            ask;           // Prezzo Ask corrente
      double            last;          // Prezzo corrente dell'ultimo trade (Last)
      ulong             volume;        // Volume per il corrente prezzo Last
      long              time_msc;      // Tempo di aggiornamento del prezzo Last in mi
      uint              flags;         // Tick flags
     };
   //--- ottiene l'ultimo valore tick
```

```
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
//--- tentativo di copiare dati da MqlTick a CustomMqlTick
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      //--- la copia di strutture semplici non correlate è vietata
      //1. my_tick1=last_tick;               // il compilatore restituisce un errore c

      //--- il typecastin di strutture non correlate l'una all'altra, è vietato, pure
      //2. my_tick1=(CustomMqlTick)last_tick;// il compilatore restituisce un errore c

      //--- quindi copia i membri della struttura uno per uno
      my_tick1.time=last_tick.time;
      my_tick1.bid=last_tick.bid;
      my_tick1.ask=last_tick.ask;
      my_tick1.volume=last_tick.volume;
      my_tick1.time_msc=last_tick.time_msc;
      my_tick1.flags=last_tick.flags;

      //--- è consentito copiare gli oggetti dello stesso tipo di CustomMqlTick nel se
      my_tick2=my_tick1;

      //--- creare un array dagli oggetti della struttura semplice CustomMqlTick e sc
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
      ArrayPrint(arr);
/ --- esempio di visualizzazione dei valori dell'array contenente gli oggetti del tip
      /*
                        [time]    [bid]    [ask]    [last] [volume]     [time_msc] [flags]
      [0] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      [1] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      */
     }
   else
      Print("SymbolInfoTick() failed, error = ",GetLastError());
  }
```

Il secondo esempio mostra le caratteristiche della copia di strutture semplici dalla discendenza. Supponiamo di avere la struttura base Animal, da cui derivano le strutture Cat e Dog. Possiamo copiare gli oggetti Animal e Cat, così come gli oggetti Animal e Dog, ma non possiamo copiare Cat e Dog a vicenda, anche se entrambi sono discendenti della struttura Animal.

```
//--- struttura per descrivere i cani
struct Dog: Animal
  {
   bool              hunting;       // razza da caccia
  };
//--- struttura per descrivere i gatti
```

```
struct Cat: Animal
  {
   bool              home;          // animale domestico
  };
//--- creare oggetti di strutture figlio
   Dog dog;
   Cat cat;
//--- può essere copiato da antenato a discendente (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // i cani possono nuotare
//--- non è possibile copiare oggetti di strutture figlie (Dog != Cat)
   cat=dog;         // il compilatore restituisce un errore
```

Codice completo di esempio:

```
//--- struttura di base per la descrizione degli animali
struct Animal
  {
   int               head;          // numero di teste
   int               legs;          // numero di zampe
   int               wings;         // numero di ali
   bool              tail;          // coda
   bool              fly;           // che vola
   bool              swim;          // che nuota
   bool              run;           // che corre
  };
//--- struttura per descrivere i cani
struct Dog: Animal
  {
   bool              hunting;       // razza da caccia
  };
//--- struttura per descrivere i gatti
struct Cat: Animal
  {
   bool              home;          // animale domestico
  };
//+------------------------------------------------------------------+
//| Funzione start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea e descrive un oggetto del tipo di base Animal
   Animal some_animal;
   some_animal.head=1;
   some_animal.legs=4;
   some_animal.wings=0;
   some_animal.tail=true;
   some_animal.fly=false;
   some_animal.swim=false;
```

```
   some_animal.run=true;
//--- crea oggetti di tipo figlio
   Dog dog;
   Cat cat;
//--- può essere copiato da antenato a discendente (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // i cani possono nuotare
//--- non è possibile copiare oggetti di strutture figlie (Dog != Cat)
   //cat=dog;        // il compilatore restituisce errore qui
//--- è pertanto possibile copiare gli elementi uno per uno, solamente
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;   // cats cannot swim
//--- è possibile copiare i valori dal discendente all'antenato
   Animal elephant;
   elephant=cat;
   elephant.run=false;// gli elefanti non possono correre
   elephant.swim=true;// gli elefanti possono nuotare
//--- crea un array
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- stampare
   ArrayPrint(animals);
//--- risultato dell' esecuzione
/*
        [head] [legs] [wings] [tail] [fly] [swim] [run]
   [0]      1      4       0   true false  false  true
   [1]      1      4       0   true false   true  true
   [2]      1      4       0   true false  false false
   [3]      1      4       0   true false   true false
*/
  }
```

Un altro modo per copiare i tipi semplici è l'utilizzo dell' unione. Gli oggetti delle strutture dovrebbero essere membri della stessa unione - vedere l'esempio in union.

L'accesso ai Membri Struttura

La struttura è un nuovo tipo di dati che consente di dichiarare variabili di questo tipo. La struttura può essere dichiarata solo una volta all'interno di un progetto. I membri della struttura sono accessibili usando l'operazione point(.).

Esempio:

```
struct trade_settings
  {
   double take;          // valori della fissazione dei prezzi profitto
   double stop;          // valore del prezzo di stop protettivo
   uchar  slippage;      // valore dello scostamento accettabile
  };
//--- creare ed inizializza una variabile di tipo trade_settings
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

'pack' per allineare i campi struttura e classe

Lo speciale l'attributo pacck consente di impostare l'allineamento dei campi della struttura o della classe.

```
 pack([n])
```

dove n è uno dei seguenti valori: 1, 2, 4, 8 o 16. Potrebbe essere assente.

Esempio:

```
   struct pack(sizeof(long)) MyStruct
     {
      // i membri della struttura devono essere allineati al limite di 8 byte
     };
or
   struct MyStruct pack(sizeof(long))
     {
      // i membri della struttura devono essere allineati al limite di 8 byte
     };
```

'pack (1)' è applicato di default per le strutture. Ciò significa che i membri della struttura si trovano uno dopo l'altro in memoria e la dimensione della struttura è uguale alla somma della dimensione dei membri.

Esempio:

```
//+------------------------------------------------------------------+
// | Funzione Start del programma di Script                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- struttura semplice senza allineamento
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- dichiara una semplice istanza di struttura
```

```
   Simple_Structure s;
//--- mostra la dimensione di ogni membro della struttura
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- assicurati che la dimensione della struttura POD sia uguale alla somma della dir
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Result:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

L'allineamento dei campi della struttura può essere necessario quando si scambiano dati con librerie di terze parti (*.DLL) in cui viene applicato tale allineamento.

Usiamo alcuni esempi per mostrare come funziona l'allineamento. Applicheremo una struttura composta da quattro membri senza allineamento.

```
//--- struttura semplice senza allineamento
   struct Simple_Structure pack() // nessuna dimensione è specificata, l'allineamento
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- dichiara una semplice istanza di struttura
   Simple_Structure s;
```

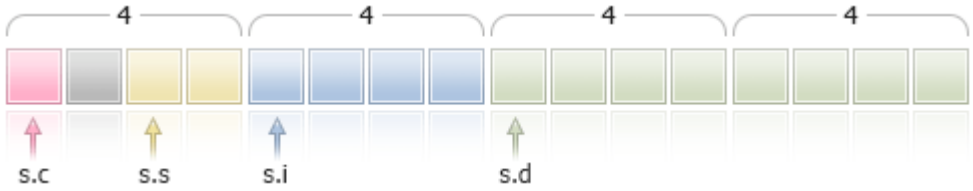
I campi della struttura devono essere collocati in memoria uno dopo l'altro in base all'ordine di dichiarazione e dimensione del tipo. La dimensione della struttura è 15, mentre un offset rispetto ai campi della struttura negli array non è definito.

Ora dichiariamo la stessa struttura con l'allineamento di 4 byte ed eseguire il codice.

```
//+------------------------------------------------------------------+
// | Funzione Start del programma di Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
//--- struttura semplice con l'allineamento a 4 byte
   struct Simple_Structure pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- dichiara una semplice istanza di struttura
   Simple_Structure s;
//--- mostra la dimensione di ogni membro della struttura
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- assicurati che la dimensione della struttura POD non sia ora uguale alla somma d
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Result:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=16 // la dimensione della struttura è cambiata
*/
  }
```

La dimensione della struttura è cambiata in modo che tutti i membri di 4 byte e più abbiano un offset dall'inizio della struttura multipla di 4 byte. I membri più piccoli devono essere allineati al loro limite di dimensione (ad esempio, 2 per "short"). Ecco come appare (il byte aggiunto è mostrato in grigio).



In questo caso, 1 byte viene aggiunto dopo il **sc** membro, in modo che il campo **s.s** (sizeof (short)==2) ha il limite di 2 byte (allineamento per il tipo 'short').

Anche l'offset all'inizio della struttura nell'array è allineato al limite di 4 byte, ovvero gli indirizzi degli elementi a [0], a[1] ed a[n] devono essere multipli di 4 byte per Simple_Structure arr[].

Consideriamo altre due strutture costituite da tipi simili con un allineamento di 4 byte ma un ordine di membro diverso. Nella prima struttura, i membri si trovano nell'ordine crescente della dimensione del tipo.

```
//+------------------------------------------------------------------+
// | Funzione Start del programma di Script                          |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
//--- semplice struttura allineata al limite di 4 byte
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- dichiara una semplice istanza di struttura
   CharShortInt ch_sh_in;
//--- mostra la dimensione di ogni membro della struttura
   Print("sizeof(ch_sh_in.c)=",sizeof(ch_sh_in.c));
   Print("sizeof(ch_sh_in.s)=",sizeof(ch_sh_in.s));
   Print("sizeof(ch_sh_in.i)=",sizeof(ch_sh_in.i));

//--- assicurati che la dimensione della struttura POD sia uguale alla somma della dir
   Print("sizeof(CharShortInt)=",sizeof(CharShortInt));
/*
  Result:
   sizeof(ch_sh_in.c)=1
   sizeof(ch_sh_in.s)=2
   sizeof(ch_sh_in.i)=4
   sizeof(CharShortInt)=8
*/
  }
```

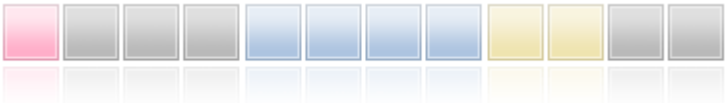
Come possiamo vedere, la dimensione della struttura è 8 e consiste dei due blocchi da 4 byte. Il primo blocco contiene i campi con i tipi "char" e "short", mentre il secondo contiene il campo con il tipo 'int'.

Ora trasformiamo la prima struttura nella seconda, che differisce solo nell'ordine dei campi, spostando il membro di tipo 'short' alla fine.

```
//+------------------------------------------------------------------+
// | Funzione Start del programma di Script                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- semplice struttura allineata al limite di 4 byte
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
     };
//--- dichiara una semplice istanza di struttura
```

```
   CharIntShort ch_in_sh;
//--- mostra la dimensione di ogni membro della struttura
   Print("sizeof(ch_in_sh.c)=",sizeof(ch_in_sh.c));
   Print("sizeof(ch_in_sh.i)=",sizeof(ch_in_sh.i));
   Print("sizeof(ch_in_sh.s)=",sizeof(ch_in_sh.s));
//--- assicurati che la dimensione della struttura POD sia uguale alla somma della dir
   Print("sizeof(CharIntShort)=",sizeof(CharIntShort));
/*
  Result:
   sizeof(ch_in_sh.c)=1
   sizeof(ch_in_sh.i)=4
   sizeof(ch_in_sh.s)=2
   sizeof(CharIntShort)=12
*/
  }
```

Sebbene il contenuto della struttura non sia cambiato, l'alterazione della sequenza dei membri ha aumentato le sue dimensioni.



L'allineamento dovrebbe anche essere considerato quando si eredita. Dimostriamolo usando la semplice struttura Parent con un singolo membro di tipo 'char'. La dimensione della struttura senza allineamento è 1.

```
   struct Parent
     {
      char              c;     // sizeof(char)=1
     };
```

Creiamo la classe figlia Children con il membro di tipo "short" (sizeof (short)=2).

```
   struct Children pack(2) : Parent
     {
      short             s;    // sizeof(short)=2
     };
```

Di conseguenza, quando si imposta l'allineamento su 2 byte, la dimensione della struttura è uguale a 4, sebbene la dimensione dei suoi membri sia 3. In questo esempio, 2 byte devono essere allocati alla classe Parent, in modo che l'accesso al campo "short" della classe figlia sia allineato a 2 byte.

La conoscenza di come viene allocata la memoria per i membri della struttura è necessaria se un'applicazione MQL5 interagisce con i dati di terze parti scrivendo/leggendo sul livello dei file o degli stream.

La directory MQL5\Include\WinAPI della Libreria standard contiene le funzioni per lavorare con le funzioni WinAPI. Queste funzioni applicano le strutture con un allineamento specificato per i casi quando è richiesto per lavorare con WinAPI.

offsetof è un comando speciale direttamente correlato all'attributo pack. Ci consente di ottenere un offset membro dall'inizio della struttura.

```
//--- dichiara la variabile di tipo Children
   Children child;
//--- rileva gli offset dall'inizio della struttura
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
  Result:
   offsetof(Children,c)=0
   offsetof(Children,s)=2
*/
```

Modificatore 'final'

L'uso del modificatore 'final' durante la dichiarazione struttura vieta ulteriori eredità da questa struttura. Se una struttura non necessita di ulteriori modifiche, o le modifiche non sono ammesse per motivi di sicurezza, dichiarare tale struttura con il modificatore 'final'. Inoltre, tutti i componenti della struttura saranno implicitamente considerati definitivi.

```
struct settings final
  {
  //--- Corpo della struttura
  };

struct trade_settings : public settings
  {
  //--- Corpo della struttura
  };
```

Se si tenta di ereditare da una struttura con il modificatore 'final', come mostrato nell'esempio precedente, il compilatore restituirà un errore:

```
non può ereditare da 'Impostazioni', siccome è stato dichiarato come 'final'
vedi dichiarazione di 'impostazioni'
```

Classi

Le classi differiscono dalle strutture in base a quanto riportato qui di seguito:

- la parola chiave class viene utilizzata nella dichiarazione;
- Per impostazione predefinita, tutti i membri della classe hanno lo specificatore di accesso private, se non diversamente indicato. Data-membri della struttura hanno il tipo predefinito di accesso come public, salvo diversa indicazione;
- oggetti della classe hanno sempre una tabella di funzioni virtuali, anche se non ci sono funzioni virtuali dichiarate nella classe. Le strutture non possono avere funzioni virtuali;
- l'operatore new può essere applicato ad oggetti della classe, questo operatore non può essere applicato a strutture;
- le classi possono essere ereditate solo dalle classi; le strutture possono essere ereditate solo dalle strutture.

Le classi e le strutture possono avere un costruttore e distruttore espliciti. Se il costruttore è definito in modo esplicito, l'inizializzazione di una struttura o variabile della classe, utilizzando la sequenza di inizializzazione, è impossibile.

Esempio:

```
struct trade_settings
  {
   double take;         // valori della fissazione dei prezzi profitto
   double stop;         // valore del prezzo di stop protettivo
   uchar  slippage;     // valore dello scostamento accettabile
   //--- Costruttore
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- Distruttore
         ~trade_settings() { Print("Questa è la fine"); }
  };
//--- Il compilatore genera un messaggio di errore riferendo che l'inizializzazione è
trade_settings my_set={0.0,0.0,5};
```

Costruttori e distruttori

Un costruttore è una funzione speciale, che viene chiamata automaticamente quando si crea un oggetto di una struttura o di una classe, e di solito è usato per inizializzare membri della classe. Inoltre parleremo solo di classi, mentre lo stesso vale per le strutture, se non diversamente indicato. Il nome di un costruttore deve corrispondere al nome della classe. Il costruttore non ha alcun tipo di ritorno (è possibile specificare il tipo void).

I membri definiti della classe - stringhe, array dinamici ed oggetti che richiedono inizializzazioni - verranno in ogni caso inizializzati, indipendentemente dal fatto che vi sia un costruttore.

Ogni classe può avere multipli costruttori, differendo per il numero di parametri e la lista di inizializzazione. Un costruttore che richiede il fatto di specificare i parametri, viene chiamato un costruttore parametrico.

Un costruttore senza parametri viene chiamato **un costruttore di default**. Se non vi sono costruttori dichiarati in una classe, il compilatore crea un costruttore di default durante la compilazione.

```
//+------------------------------------------------------------------------------+
//| La classe per lavorare con una data                                          |
//+------------------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;          // Anno
   int               m_month;         // Mese
   int               m_day;           // Giorno del mese
   int               m_hour;          // Ore in un giorno
   int               m_minute;        // Minuti
   int               m_second;        // Secondi
public:
   //--- Costruttore default
```

```
                     MyDateClass(void);
   //--- Costruttore parametrico
                     MyDateClass(int h,int m,int s);
  };
```

Un costruttore può essere dichiarato nella descrizione della classe e poi il suo corpo può essere definito. Ad esempio, due costruttori di MyDateClass possono essere definiti nel modo seguente:

```
//+------------------------------------------------------------------------------------+
//| Costruttore default                                                                |
//+------------------------------------------------------------------------------------+
MyDateClass::MyDateClass(void)
  {
//---
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=mdt.hour;
   m_minute=mdt.min;
   m_second=mdt.sec;
   Print(__FUNCTION__);
  }
//+------------------------------------------------------------------------------------+
//| Costruttore parametrico                                                            |
//+------------------------------------------------------------------------------------+
MyDateClass::MyDateClass(int h,int m,int s)
  {
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=h;
   m_minute=m;
   m_second=s;
   Print(__FUNCTION__);
  }
```

Nel costruttore di default, tutti i membri della classe vengono riempiti con la funzione TimeCurrent(), nel costruttore parametrico solo i valori orari vengono compilati Altri membri della classe (m_year, m_month e m_day) viene automaticamente inizializzati con la data corrente.

Il costruttore predefinito ha uno scopo speciale durante l'inizializzazione di un array di oggetti della sua classe. Il costruttore, di cui tutti i parametri hanno valori predefiniti, **non è** un costruttore di default. Ecco un esempio:

```
//+------------------------------------------------------------------------------------+
```

```
//| La classe per il costruttore di default                                  |
//+------------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // Ora della chiamata ultimo oggetto
public:
   //--- Un costruttore con un parametro che ha un valore predefinito non è un costrut
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- Un costruttore di copia
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                   |
//+------------------------------------------------------------------------+
voidOnStart()
  {
// CFoo foo; // Questa variante non può essere utilizzata - un costruttore di default
//--- Le possibili opzioni per creare l'oggetto CFoo
   CFoo foo1(TimeCurrent());      // Una chiamata esplicita di un costruttore parametri
   CFoo foo2();                   // Una chiamata esplicita di un costruttore parametri
   CFoo foo3=D'2009.09.09';       // Una chiamata implicita di un costruttore parametri
   CFoo foo40(foo1);              // Una chiamata esplicita di un costruttore di copia
   CFoo foo41=foo1;               // Una chiamata implicita di un costruttore di copia
   CFoo foo5;                     // Una chiamata esplicita di un costruttore di defaul
                                  // allora viene chiamato un costruttore parametrico
//--- Le opzioni possibili per ricevere puntatori CFoo
   CFoo *pfoo6=new CFoo();        // Creazione dinamica di un oggetto e la ricezione di
   CFoo *pfoo7=new CFoo(TimeCurrent());// Un'altra opzione di creazione di oggetti din
   CFoo *pfoo8=GetPointer(foo1); // Ora pfoo8 punta all'oggetto foo1
   CFoo *pfoo9=pfoo7;             // pfoo9 e pfoo7 puntano all'unico e stesso oggetto
   // CFoo foo_array[3];          // Questa opzione non può essere utilizzata - un cost
//--- Mostra il valore di m_call_time
   Print("foo1.m_call_time=",foo1.ToString());
   Print("foo2.m_call_time=",foo2.ToString());
   Print("foo3.m_call_time=",foo3.ToString());
   Print("foo4.m_call_time=",foo4.ToString());
   Print("foo5.m_call_time=",foo5.ToString());
   Print("pfoo6.m_call_time=",pfoo6.ToString());
   Print("pfoo7.m_call_time=",pfoo7.ToString());
   Print("pfoo8.m_call_time=",pfoo8.ToString());
   Print("pfoo9.m_call_time=",pfoo9.ToString());
//--- Elimina array creati dinamicamente
   delete pfoo6;
   delete pfoo7;
   //delete pfoo8;  // Non è necessario eliminare pfoo8 in modo esplicito, in quanto
   //delete pfoo9;  // Non è necessario eliminare pfoo9 esplicitamente. in quanto punt
  }
```

Se si toglie il commento alle stringhe

```
   //CFoo foo_array[3];     // Questa variante non può essere utilizzata - un costrutto
```

o

```
   //CFoo foo_dyn_array[];  // Questa variante non può essere utilizzata - un costrutto
```

il compilatore restituisce un errore per loro "il costruttore di default non è definito".

Se una classe ha un costruttore definito-dall'utente, il costruttore di default non viene generato dal compilatore. Ciò significa che se un costruttore parametrico è dichiarato in una classe, ma un costruttore di default non è dichiarato, non è possibile dichiarare gli array di oggetti di questa classe. Il compilatore restituirà un errore per questo script:

```
//+------------------------------------------------------------------------------+
//| Una classe senza un costruttore di default                                   |
//+------------------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Ottiene durante la compilazione l'errore "il costruttore di default non è defini
   CFoo badFoo[5];
  }
```

In questo esempio, la classe CFoo dispone di un costruttore parametrico dichiarato - in questi casi, il compilatore non crea un costruttore di default automaticamente durante la compilazione. Allo stesso tempo, quando si dichiara un array di oggetti, si presume che tutti gli oggetti debbano essere creati ed inizializzati automaticamente. Durante l'auto-inizializzazione di un oggetto, è necessario chiamare un costruttore di default, ma dal momento che il costruttore di default non è esplicitamente dichiarato e non generato automaticamente dal compilatore, non è possibile creare un tale oggetto. Per questo motivo, il compilatore genera un errore in fase di compilazione.

C'è una sintassi speciale per inizializzare un oggetto utilizzando un costruttore. Inizializzatori Costruttore (costruzioni speciali per l'inizializzazione) per i membri di una struttura o classe, possono essere specificato nella lista di inizializzazione.

Un elenco di inizializzazione è un elenco di inizializzatori separati da virgole, che arriva dopo i due punti dopo la lista dei parametri di un costruttore e precede il corpo (va prima di una parentesi graffa di apertura). Ci sono diversi requisiti:

- Elenchi di inizializzazione possono essere utilizzati solo nei costruttori;
- Genitore membri non possono essere inizializzati nella lista di inizializzazione;

- L'elenco di inizializzazione deve essere seguito da una definizione (implementazione) di una funzione.

Ecco un esempio di diversi costruttori per inizializzare i membri della classe.

```
//+------------------------------------------------------------------------------+
//| Una classe per memorizzare il nome di un personaggior                        |
//+------------------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // Primo nome
   string            m_second_name;    // Secondo nome
public:
   //--- Un costruttore di default vuoto
                     CPerson() {Print(__FUNCTION__);};
   //--- Un costruttore parametrico
                     CPerson(string full_name);
   //--- Un costruttore con un elenco di inizializzazione
                     CPerson(string surname,string name): m_second_name(surname), m_fi
   void PrintName(){PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
  };
//+------------------------------------------------------------------------------+
//|                                                                              |
//+------------------------------------------------------------------------------+
CPerson::CPerson(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>=0)
     {
      m_first_name=StringSubstr(full_name,0,pos);
      m_second_name=StringSubstr(full_name,pos+1);
     }
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Riceve un errore "il costruttore di default non è definito"
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                        // Tom Sawyer
   CPerson Huck("Huckleberry","Finn");              // Huckleberry Finn
   CPerson *Pooh = new CPerson("Winnie","Pooh");  // Winnie the Pooh
   //--- Valori in Output
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- Eliminazione di un oggetto creato in modo dinamico
   delete Pooh;
```

```
   }
```

In questo caso, la classe CPerson ha tre costruttori:

1. Un esplicito costruttore di default, che permette di creare un array di oggetti di questa classe;
2. Un costruttore con un parametro, che ottiene un nome completo come parametro e lo divide per il nome e il secondo nome secondo lo spazio trovato;
3. Un costruttore con due parametri che contiene un elenco di inizializzazione. Inizializzatori - m_second_name(surname) e m_first_name(name).

Si noti che l'inizializzazione utilizzando una lista ha sostituito un' assegnazione. I singoli membri devono essere inizializzati come:

```
class_member (una lista di espressioni)
```

Nella lista di inizializzazione, i membri possono andare in qualsiasi ordine, ma tutti i membri della classe verranno inizializzati secondo l'ordine del loro annuncio. Ciò significa che nel terzo costruttore, prima il membro m_first_name verrà inizializzato, come è annunciato prima, e solo dopo che m_second_name viene inizializzato. Ciò deve essere tenuto in considerazione nel caso in cui l'inizializzazione di alcuni membri della classe dipendono dai valori di altri membri della classe.

Se un costruttore di default non viene dichiarato nella classe base, ed al tempo stesso vengono dichiarati uno o più costruttori con parametri, si dovrebbe sempre chiamare uno dei costruttori della classe base nella lista di inizializzazione. Passa attraverso la virgola come membri ordinari della lista e verrà chiamato la prima volta durante l'inizializzazione dell'oggetto, non importa dove si trova nell'elenco di inizializzazione.

```
//+------------------------------------------------------------------------------+
//| Classe Base                                                                  |
//+------------------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- Un costruttore con un elenco di inizializzazione
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------------------+
//| Classe derivata da CFoo                                                      |
//+------------------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;      // Un membro della classe è un oggetto del genitor
public:
   //--- Un costruttore di default nella lista di inizializzazione chiama il costrutto
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
```

```
  {
   CBar bar;
  }
```

In questo esempio, quando si crea l'oggetto bar, verrà chiamato un costruttore di default CBar(), in cui prima viene chiamato un un costruttore per la CFoo padre, e poi arriva un costruttore per il membro della classe m_member.

Un distruttore è una funzione speciale che viene chiamata automaticamente quando un oggetto della classe viene distrutto. Il nome del distruttore è scritto come il nome della classe con una tilde (~). Stringhe, array dinamici ed oggetti, richiedenti la deinizializzazione, verranno de-inizializzati in ogni caso, indipendentemente dalla presenza o assenza del distruttore. Se c'è un distruttore, queste azioni verranno eseguite dopo aver chiamato il distruttore.

I distruttori sono sempre virtuali, indipendentemente dal fatto che siano dichiarati con la parola chiave virtual o meno.

Definizione dei Metodi della Classe

I metodi-funzione della classe possono essere definiti sia all'interno della classe che all'esterno della dichiarazione della classe. Se il metodo viene definito all'interno di una classe, allora il suo corpo viene subito dopo la dichiarazione del metodo.

Esempio:

```
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
   void              Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                     { return;                }
   virtual bool      CheckDown(int& pad_array[]);
```

```
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Le funzioni da SetRightBorder (confine int) per Draw() vengono dichiarate e definite direttamente all'interno della classe CTetrisShape.

Il costruttore CTetrisShape() ed i metodi CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) and CheckRight(int& side_row[]) vengono solo dichiarati all'interno della classe, ma non ancora definiti. Le definizioni di queste funzioni ci saranno ulteriormente, nel codice. Per definire il metodo all'esterno della classe, viene utilizzato l' operatore di risoluzione ambito, il nome della classe viene utilizzato come ambito.

Esempio:

```
//+------------------------------------------------------------------------------------+
//| Costruttore della classe base                                                      |
//+------------------------------------------------------------------------------------+
void CTetrisShape::CTetrisShape()
  {
   m_type=0;
   m_ypos=0;
   m_xpos=0;
   m_xsize=SHAPE_SIZE;
   m_ysize=SHAPE_SIZE;
   m_prev_turn=0;
   m_turn=0;
   m_right_border=0;
  }
//+------------------------------------------------------------------------------------+
//| Controlla l'abilità di spostarsi giù (per il "lungo" ed il cubo)                   |
//+------------------------------------------------------------------------------------+
bool CTetrisShape::CheckDown(int& pad_array[])
  {
   int i,xsize=m_xsize/SHAPE_SIZE;
//---
   for(i=0; i<xsize; i++)
     {
      if(m_ypos+m_ysize>=pad_array[i]) return(false);
     }
//---
   return(true);
  }
```

Modificatori di accesso Public, Protected e Private

Nello sviluppo di una nuova classe, si consiglia di limitare l'accesso ai membri dall'esterno. Per questo scopo vengono utilizzate le parole chiave private oprotected. In questo caso, i dati nascosti possono essere acceduti solo dai metodi-funzione della stessa classe. Se viene utilizzata la parola chiave

protected, i dati nascosti possono essere acceduti anche dai metodi delle classi - eredi di questa classe. Lo stesso metodo può essere utilizzato per limitare l'accesso ai metodi-funzione di una classe.

Se è necessario aprire completamente l'accesso ai membri e/o metodi di una classe, utilizzare la parola chiave public.

Esempio:

```
class CTetrisField
  {
private:
   int               m_score;                           // Punteggio
   int               m_ypos;                            // Posizione corrente delle
   int               m_field[FIELD_HEIGHT][FIELD_WIDTH]; // Matrice del pozzo
   int               m_rows[FIELD_HEIGHT];              // Numerazione delle righe de
   int               m_last_row;                        // Ultime righe libere
   CTetrisShape     *m_shape;                           // Figura Tetris
   bool              m_bover;                           // Game over
public:
   void              CTetrisField() { m_shape=NULL; m_bover=false; }
   void              Init();
   void              Deinit();
   void              Down();
   void              Left();
   void              Right();
   void              Rotate();
   void              Drop();
private:
   void              NewShape();
   void              CheckAndDeleteRows();
   void              LabelOver();
  };
```

I membri della classe e metodi dichiarati dopo l'identificatore **public:** (e prima dello specificatore di accesso successivo) sono disponibili in qualsiasi riferimento all'oggetto classe dal programma. In questo esempio questi sono i seguenti membri: funzioni CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() and Drop().

I membri che vengono dichiarati dopo lo specificatore di accesso agli elementi **private:** (e prima dello specificatore di accesso successivo) sono disponibili solo per i membri funzioni di questa classe. Gli secificatori di accesso agli elementi finiscono sempre con i due punti (:) e possono comparire nella definizione della classe molte volte.

Qualsiasi membro della classe dichiarato dopo l' identificatore di accesso protected:(e fino al successivo identificatore di accesso) sono disponibili solo per le funzioni-membro di questa classe e per le funzioni-membro delle classi discendenti. Quando si tenta di fare riferimento ai membri che presentano gli specificatori private e protected dall'esterno, otteniamo l'errore della fase di compilazione. Esempio:

```
class A
  {
```

```
protected:
// --- l'operatore copy è disponibile solo all'interno della classe A e suoi discender
   void operator=(const A &)
     {
     }
  };
class B
  {
   //--- oggetto della classe A dichiarato
   A                 a;
  };
//+------------------------------------------------------------------+
//| Funzione di avvio del programma di script                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara due variabili di tipo B.
   B b1, b2;
//--- tenta di copiare un oggetto in un altro
   b2=b1;
  }
```

Durante la compilazione del codice, viene visualizzato il messaggio di errore - tentativo di chiamare l'operatore copy remoto:

```
attempting to reference deleted function 'void B::operator=(const B&)'   trash3.mq5
```

La seconda stringa di seguito fornisce una descrizione più dettagliata — l'operatore copy nella classe B è stato eliminato esplicitamente, poiché viene chiamato l'operatore copy non disponibile della classe A:

```
   function 'void B::operator=(const B&)' was implicitly deleted because it invokes ir
```

L'accesso ai membri della classe base può essere ridefinito durante[eredità](#) nelle classi derivate.

'delete' specifier

L' identificatore delete contrassegna le funzioni dei membri della classe che non possono essere utilizzate. Ciò significa che se il programma fa riferimento a tale funzione in modo esplicito o implicito, l'errore viene già ricevuto in fase di compilazione. Ad esempio, questo identificatore consente di rendere i metodi parent non disponibili in una classe child. Lo stesso risultato può essere ottenuto se dichiariamo la funzione nell'area privata della classe genitrice (dichiarazioni nella sezione private). Qui, usare delete rende il codice più leggibile e gestibile a livello di discendenti.

```
class A
  {
public:
                    A(void) {value=5;};
   double           GetValue(void) {return(value);}
private:
   double           value;
  };
```

```
class B: public A
  {
   double            GetValue(void)=delete;
  };
//+------------------------------------------------------------------+
//| Funzione di avvio del programma di script                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara la variabile di tipo A.
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- tenta di ottenere valore dalla variabile di tipo B.
   B b;
   Print("b.GetValue()=", b.GetValue()); // il compilatore mostra un errore in questa
  }
```

Il messaggio del compilatore:

```
attempting to reference deleted function 'double B::GetValue()'
   function 'double B::GetValue()' was explicitly deleted here
```

Lo specificatore 'delete' consente di disabilitare il casting automatico o il costruttore copy, che altrimenti dovrebbe essere nascosto nella sezione private anche. Esempio:

```
class A
  {
public:
   void              SetValue(double v) {value=v;}
//--- disabilita la chiamata di tipo int
   void              SetValue(int) = delete;
//--- disabilita l'operatore copy
   void              operator=(const A&) = delete;
private:
   double            value;
  };
//+------------------------------------------------------------------+
//| Funzione di avvio del programma di script                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara due variabili di tipo A.
   A a1, a2;
   a1.SetValue(3);      // errore!
   a1.SetValue(3.14);   // OK
   a2=a1;               // errore!
  }
```

Durante la compilazione, riceviamo i messaggi di errore:

```
tentativodiriferimento ad una funzione 'void cancellataUna::funzione SetValue(int)'
```

```
    'void A::SetValue(int)' was explicitly deleted here
attempting to reference deleted function 'void A::operator=(const A&)'
   function 'void A::operator=(const A&)' was explicitly deleted here
```

Modificatore 'final'

L'uso del modificatore 'final' durante la dichiarazione della classe vieta ulteriori eredità da questa classe. Se l'interfaccia della classe non richiede ulteriori modifiche, o le modifiche non sono ammesse per motivi di sicurezza, dichiarare questa classe con il modificatore 'final'. Inoltre, tutti i membri della classe saranno implicitamente considerati definitivi.

```
class CFoo final
  {
  //--- Corpo della classe
  };

class CBar : public CFoo
  {
  //--- Corpo della classe
  };
```

Se si tenta di ereditare da una classe con il modificatore 'final', come mostrato nell'esempio precedente, il compilatore restituirà un errore:

```
non si può ereditare da 'CFoo' siccome è stato dichiarato come 'final'
vedi dichiarazione di 'CFoo'
```

Unioni (union)

Union è un tipo di dati speciale costituito da diverse variabili che condividono la stessa area di memoria. Pertanto, l'unione fornisce la capacità di interpretare la stessa sequenza di bit in due (o più) modi diversi. La dichiarazione dell'unione è simile alla dichiarazione della struttura ed inizia con la parola chiave union.

```
union LongDouble
{
  long   long_value;
  double double_value;
};
```

A differenza della struttura, i vari membri sindacali appartengono alla stessa area di memoria. In questo esempio, l'unione di LongDouble è dichiarata con valori di tipo long e double che condividono la stessa area di memoria. Si prega di notare che è impossibile fare in modo che union memorizzi (a differenza di una struttura) simultaneamente valori integer *long* e valori real *double* , poiché le variabili long_value e double_value si sovrappongono (in memoria). D'altro canto, un programma MQL5 è in grado di elaborare in qualsiasi momento i dati contenenti nell'unione un valore intero(long) o reale(double). Pertanto, l'unione consente di ricevere due (o più) opzioni per rappresentare la stessa sequenza di dati.

Durante la dichiarazione dell' unione, il compilatore assegna automaticamente l'area di memoria sufficiente per memorizzare il tipo più grande (per volume) nella variabile union. La stessa sintassi viene utilizzata per accedere all'elemento union come per le strutture - operatore point.

```
union LongDouble
```

```
{
  long   long_value;
  double double_value;
};
//+------------------------------------------------------------------+
//| Funzione start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   LongDouble lb;
//--- ottiene e visualizza il numero -nan(ind) non valido
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%I64X",lb.double_value,lb.long_value);
//--- il più ampio valore normalizzato (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- il più piccolo positivo normalizzato (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/* Risultato dell'esecuzione
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

Poiché le unioni consentono al programma di interpretare gli stessi dati di memoria in modi diversi, esse vengono spesso utilizzate quando un' inusuale conversione di tipo è richiesta.

Le unioni non possono essere coinvolte nella eredità(inheritance), e loro anche non possono avere membri statici a causa della loro stessa natura. In tutti gli altri aspetti, l' *unione* si comporta come una struttura con tutti i suoi membri con un offset di zero. I seguenti tipi non possono essere i membri dell'unione:

- dynamic arrays
- strings
- puntatori ad oggetti e funzioni
- oggetti della classe
- oggetti di struttura con costruttori o distruttori
- oggetti di struttura aventi membri dai punti 1-5

Simillmente alle classi, l'unione è in grado di avere costruttori e distruttori, così come metodi. Per default, i membri dell'unione sono di tipo di accesso public. Per creare elementi privati, utilizzare la parola chiave private. Tutte queste possibilità sono visualizzate nell'esempio che illustra come convertire un colore del tipo color in ARGB come fa la funzione ColorToARGB().

```
//+------------------------------------------------------------------+
//| Unione per la conversione in colori (BGR) in ARGB |
//+------------------------------------------------------------------+
```

```
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- costruttori
                     ARGB(color col,uchar a=0){Color(col,a);};
                    ~ARGB(){};
   //--- metodi pubblici
public:
   uchar   Alpha(){return(argb[3]);};
   void    Alpha(const uchar alpha){argb[3]=alpha;};
   color   Color(){ return(color(clr));};
   //--- metodi privati
private:
   //+------------------------------------------------------------------+
   //| imposta il valore del canale alpha ed il colore |
   //+------------------------------------------------------------------+
   void    Color(color col,uchar alpha)
     {
      //--- imposta il colore al membro clr
      clr=col;
      //--- imposta il valore del componente Alpha - livello d'opacità
      argb[3]=alpha;
      //--- scambia i bite dei componenti R e B (Red e Blue)
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+------------------------------------------------------------------+
//| Funzione start programma Script                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 0x55 significa 55/255=21.6 % (0% - completamente trasparente)
   uchar alpha=0x55;
//--- il tipo di colore è rappresentato come 0x00BBGGRR
   color test_color=clrDarkOrange;
//--- qui vengono accettati i valori di byte provenienti dall'unione ARGB
   uchar argb[];
   PrintFormat("0x%.8X - qui è come il tipo 'color' appare per %s, BGR=(%s)",
               test_color,ColorToString(test_color,true),ColorToString(test_color));
//--- Il tipo ARGB è rappresentato da 0x00RRGGBB, RR e BB vengono scambiati
   ARGB argb_color(test_color);
//--- copia l'array di byte
   ArrayCopy(argb,argb_color.argb);
//--- ecco come appare nella rappresentazione ARGB
   PrintFormat("0x%.8X - rappresentazione ARGB con il canale alpha=0x%.2x, ARGB=(%d,%
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- aggiungi livello opacità
   argb_color.Alpha(alpha);
```

```
//--- prova a definire ARGB come tipo 'color'
   Print("ARGB как color=(",argb_color.clr,")  alpha channel=",argb_color.Alpha());
//--- copia l'array di byte
   ArrayCopy(argb,argb_color.argb);
//--- ecco come appare nella rappresentazione ARGB
   PrintFormat("0x%.8X - rappresentazione ARGB con il canale alpha=0x%.2x, ARGB=(%d,%d
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- controlla i risultati della funzione ColorToARGB()
   PrintFormat("0x%.8X - risultato di ColorToARGB(%s,0x%.2x)",ColorToARGB(test_color,a
               ColorToString(test_color,true),alpha);
  }
/* Risultato dell'esecuzione
   0x00008CFF - qui è come il tipo color appare per clrDarkOrange, BGR=(255,140,0)
  0x00FF8C00 - rappresentazione ARGB con il canale alpha=0x00, ARGB=(0,255,140,0)
   ARGB as color=(0,140,255)  alpha channel=85
   0x55FF8C00 - rappresentazione ARGB con il canale alfa = 0x55, ARGB = (85,255,140,0)
   0x55FF8C00 - result of ColorToARGB(clrDarkOrange,0x55)
*/
```

Interfacce

Un'interfaccia consente la determinazione di specifiche funzionalità, che una classe può quindi implementare. Infatti, un'interfaccia è una classe che non può contenere membri, e non può avere un costruttore e/o un distruttore. Tutti i metodi dichiarati in una interfaccia sono puramente virtuali, anche in assenza di una definizione esplicita.

Un'interfaccia è definita utilizzando la parola chiave "interfaccia". Esempio:

```
//--- Interfaccia base per descrivere animali
interface IAnimal
  {
//--- I metodi dell'interfaccia hanno accesso pubblico per default
   void Sound();  // Il suono prodotto dall'animale
  };
//+------------------------------------------------------------------+
//|  The classe CCat è ededitata dalla classe IAnimal interface      |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
                     CCat() { Print("Il gatto è nato"); }
                    ~CCat() { Print("Il gatto è morto");  }
   //--- Implementare il metodo Sound di IAnimal interface
   void Sound(){ Print("meou"); }
  };
//+------------------------------------------------------------------+
//|  La classe CDog è ereditata dalla classe IAnimal interface       |
//+------------------------------------------------------------------+
class CDog : public IAnimal
```

```
  {
public:
                     CDog() { Print("Il cane è nato"); }
                    ~CDog() { Print("Il cane è morto");  }
   //--- Implementare il metodo Sound di IAnimal interface
   void Sound(){ Print("guaf"); }
  };
//+------------------------------------------------------------------+
//| Programma Script funzione start                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Un array di puntatori a oggetti del tipo IAnimal
   IAnimal *animals[2];
//--- La creazione di classi figlio di IAnimal e risparmio di puntatori a loro, in un
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- Chiamata al metodo Suono() dell'interfaccia base IAnimal per ogni bambino
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- Eliminazione oggetti
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- Risultato dell'esecuzione
/*
  Il gatto è nato
  Il cane è nato
   meou
   guaf
  Il gatto è morto
  Il cane è morto
*/
  }
```

Come le classi astratte, un oggetto interfaccia non può essere creato senza eredità. Un'interfaccia può essere ereditata solo dalle altre interfacce e può essere un genitore per una classe. Un interfaccia è sempre con visibilità pubblica.

Un'interfaccia non può essere dichiarata all'interno di una dichiarazione della classe o di una struttura, ma un puntatore per l'interfaccia può essere salvato in una variabile di tipo void *. In generale, un puntatore ad un oggetto di qualsiasi classe può essere salvato in una variabile di tipo void *. Per convertire un puntatore void * a un puntatore ad un oggetto di una classe particolare, utilizzare l'operatore dynamic_cast. Se la conversione non è possibile, il risultato dell'operazione dynamic_cast sarà NULL.

Vedi anche

Programmazione Ad Oggetti (OOP Object-Oriented-Programming)

Oggetto Array Dinamico

Array dinamici

Massimo array a 4-dimensioni possono essere dichiarati. Quando si dichiara un array dinamico (un array di valore non specificato nella prima coppia di parentesi quadre), il compilatore crea automaticamente una variabile della struttura di cui sopra (un oggetto array dinamico) e fornisce un codice per la corretta inizializzazione.

Array dinamici vengono automaticamente liberati quando si va oltre l'area visibilità del blocco in cui sono dichiarati

Esempio:

```
double matrix[][10][20]; // array dinamico tri-dimensionale
ArrayResize(matrix,5);   // Imposta la grandezza della prima dimensione
```

Array statici

Quando tutte le significative dimensioni dell'array sono specificate in modo esplicito, il compilatore pre-alloca la dimensione necessaria della memoria. Un tale array è chiamato statico. Tuttavia, il compilatore alloca memoria aggiuntiva per l'oggetto di un array dinamico, il cui (oggetto) è associato al buffer statico pre-assegnato (parte della memoria per memorizzare l'array).

La creazione di un oggetto array dinamico è dovuta alla possibile necessità di passare questo array statico come parametro per qualche funzione.

Esempi:

```
double stat_array[5]; // array statico uni-dimensionale
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0) return(false);
   ...
   return(true);
  }
```

Array in Strutture

Quando un array statico è dichiarato come un membro di una struttura, un oggetto array dinamico non viene creato. Questo viene fatto per garantire la compatibilità delle strutture di dati utilizzati per la API di Windows.

Tuttavia, gli array statici che vengono dichiarati come membri delle strutture, possono anche essere passati funzioni-MQL5. In questo caso, quando si passa il parametro di un oggetto temporaneo di un array dinamico, collegato con l'array statico - verrà creato il membro della struttura.

Vedi anche

Funzioni di Array, Inizializzazione delle Variabili, Visibilità Ambito di Applicazione e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Matrices and vectors

Type vector is a special data type in MQL5, which enables operations with vectors. A vector is a one-dimensional array of type double. It is one of the fundamental concepts of linear algebra, which is used in many fields of science, including physics, geometry, and others. Vectors are used to solve systems of linear equations, in 3D graphics and in other application areas. Vectors can be added and multiplied. The length or distance between vectors can be obtained through the Norm. In programming, vectors are usually represented by arrays of homogeneous elements, which may have no regular vector operations, i.e. when arrays cannot be added or multiply, and they have no norm.

Vectors can be represented as row vectors and string vectors when working with matrices. Also, vectors in linear algebra use the concepts of covariance and contravariance. These concepts do not make any difference when writing an MQL5 code, as only the programmer decides what each object of the vector type is. For example, it can be rotation, displacement or compression vector in 3D graphics.

Generally speaking, from the point of view of linear algebra, a number is also a vector, but in a one-dimensional vector space. A vector itself can be considered as a special case of a matrix.

Type matrix is another special data type in MQL5 to represent matrices. A matrix is actually a two-dimensional array of type double. Vectors and matrices have been introduced into MQL5 for easier operations with certain types of data sets. With them, developers can benefit from the linear algebra possibilities in a simple and math-like form. Matrices can be used to compactly write systems of linear or differential equations. The number of matrix rows corresponds to the number of equations, while the the number of columns is equal to the number of unknowns. As a result, systems of linear equations can be solved through matrix operations.

The following algebraic operations are defined for the matrices:

- Addition of same-size matrices
- Multiplication of suitable-size matrices: the number of columns in the left matrix must equal the number of rows in the right matrix
- Matrix multiplication by a column vector; multiplication of a row vector by a matrix according to the matrix multiplication rule. In this sense the vector is a special case of a matrix
- Matrix multiplication by a number, that is, by a scalar

Mathematics considers many different matrix types. For example, identity matrix, symmetric, skew-symmetric, upper and lower triangular matrices, and other types. Various Normal forms play an important role in the matrix theory. They represent a certain canonical form of a matrix, which can be obtained by means of certain transformations. In practice, normal forms that have additional properties, such as for example stability, are used.

The use of vectors and matrices, or rather, of special methods of the relevant types, enables the creation of simpler, briefer and clearer code, which is close to mathematical notation. With these methods, you can avoid the need to create nested loops or to mind correct indexing of arrays in calculations. Therefore, the use of these methods increases reliability and speed in developing complex programs.

List of matrix and vector methods

Types matrix and vector include methods that correspond to the relevant NumPy library methods. Using these methods, you can translate algorithms and codes from Python to MQL5 with minimum efforts. A lot of data processing tasks, mathematical equations, neural networks and machine learning tasks can be solved using ready-made Python methods and libraries.

Method matrix/vector	Analogous method in NumPy	Description
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	Construct a matrix with ones on the diagonal and zeros elsewhere
void matrix.Identity(const int rows)	identity	Construct a square matrix with ones on the main diagonal
void matrix.Ones(const int rows, const int cols)	ones	Construct a new matrix of given rows and columns, filled with ones
void matrix.Zeros(const int rows, const int cols)	zeros	Construct a new matrix of given rows and columns, filled with zeros
void matrix.Full(const int rows, const int cols, const scalar value)	full	Construct a new matrix of given rows and columns, filled with scalar value
void matrix.Copy(const matrix a)	copy	Construct a copy of the given matrix
void matrix.FromBuffer(const int rows, const int cols, const scalar array[], const int count=-1, const int offset=0)	frombuffer	Construct a matrix created from a 1-dimensional array
void matrix.FromFile(const int rows, condt int cols, const int file_handle, const int count=-1, const int offset=0)	fromfile	Construct a matrix from data in a text or binary file
void vector.FromString(const string source, const string sep=" ")	fromstring	Construct a vector initialized from text data in a string
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	Construct evenly spaced values within a given interval
void matrix.Diag(const vector v, const int ndiag=0)	diag	Extract a diagonal or construct a diagonal matrix
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	Construct a matrix with ones at and below the given diagonal and zeros elsewhere
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	Return a copy of a matrix with elements above the k-th diagonal zeroed
void matrix.Triu(const int rows, const int cols, const scalar array[],	triu	Return a copy of a matrix with the elements below the k-th diagonal

Method matrix/vector	Analogous method in NumPy	Description
const int ndiag=0)		zeroed
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	Generate a Vandermonde matrix
vector matrix.Row(const unsigned nrow)		Return a row vector
vector matrix.Col(const unsigned ncol)		Return a column vector
unsigned matrix.Rows()		Return the number of rows in a matrix
unsigned matrix.Cols()		Return the number of columns in a matrix
void matrix.Init()		Initialize a matrix
matrix matrix.Transpose()	transpose	Reverse or permute the axes of a matrix; returns the modified matrix
matrix matrix.Dot(const matrix b)	dot	Dot product of two matrices
matrix matrix.Inner(const matrix b)	inner	Inner product of two matrices
matrix matrix.Outer(const matrix b)	outer	Compute the outer product of two matrices
matrix matrix.MatMul(const matrix b)	matmul	Matrix product of two matrices
matrix matrix.MatrixPower(const int power)	matrix_power	Raise a square matrix to the (integer) power n
matrix matrix.Kron(const matrix b)	kron	Return Kronecker product of two matrices
bool matrix.Cholesky(matrix& L)	cholesky	Return the Cholesky decomposition
bool matrix.QR(matrix& Q, matrix& R)	qr	Compute the qr factorization of a matrix
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	Singular value decomposition
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	Compute the eigenvalues and right eigenvectors of a square matrix
bool matrix.EigH(matrix& eigen_vectors, vector& eigen_values)	eigh	Return the eigenvalues and eigenvectors of a Hermitian matrix

Method matrix/vector	Analogous method in NumPy	Description
bool matrix.EigVals(vector& eigen_values)	eigvals	Compute the eigenvalues of a general matrix
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	Compute the eigenvalues of a Hermitian matrix
bool matrix.LU(matrix& L, matrix& U)		LU decomposition of a matrix as the product of a lower triangular matrix and an upper triangular matrix
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		LUP decomposition with partial pivoting, which refers to LU decomposition with row permutations only: PA=LU
double matrix.Norm(const norm)	norm	Return matrix or vector norm
double matrix.Cond(const norm)	cond	Compute the condition number of a matrix
vector matrix.Spectrum()		Compute spectrum of a matrix as the set of its eigenvalues from the product AT*A
double matrix.Det()	det	Compute the determinant of an array
int matrix.Rank()	matrix_rank	Return matrix rank of array using the Gaussian method
int matrix.SLogDet(int& sign)	slogdet	Compute the sign and logarithm of the determinant of an array
double matrix.Trace()	trace	Return the sum along diagonals of the matrix
vector matrix.Solve(const vector b)	solve	Solve a linear matrix equation, or system of linear algebraic equations
vector matrix.LstSq(const vector b)	lstsq	Return the least-squares solution of linear algebraic equations (for non-square or degenerate matrices)
matrix matrix.Inv()	inv	Compute the (multiplicative) inverse of a matrix
matrix matrix.PInv()	pinv	Compute the pseudo-inverse of a matrix by the Moore-Penrose method
int matrix.Compare(const matrix matrix_c, const double epsilon)		Confronta gli elementi di due matrici/vettori con la precisione specificata

Method matrix/vector	Analogous method in NumPy	Description
int matrix.Compare(const matrix matrix_c, const int digits) int vector.Compare(const vector vector_c, const double epsilon) int vector.Compare(const vector vector_c, const int digits)		
double matrix.Flat(const ulong index) bool matrix.Flat(const ulong index,const double value)	flat	Consente di indirizzare un elemento di matrice tramite un indice anziché due
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	Restituisce l'indice del valore massimo
double vector.ArgMin() double matrix.ArgMin() vector matrix.ArgMin(const int axis)	argmin	Restituisce l'indice del valore minimo
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	Restituisce il valore massimo in una matrice/vettore
double vector.Mean() double matrix.Mean() vector matrix.Mean(const int axis)	mean	Calcola la media aritmetica dei valori degli elementi
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	Restituisce il valore minimo in una matrice/vettore
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	Restituisce la somma degli elementi matrice/vettore che può essere eseguita anche per l'asse dato (assi).
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	Limita gli elementi di una matrice/vettore a un intervallo specificato di valori validi
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	Restituisce il prodotto cumulativo di elementi matrice/vettoriali, inclusi quelli lungo l'asse specificato
vector vector.CumSum() vector matrix.CumSum()	cumsum	Restituisce la somma cumulativa di elementi matrice/vettore, inclusi quelli lungo l'asse specificato

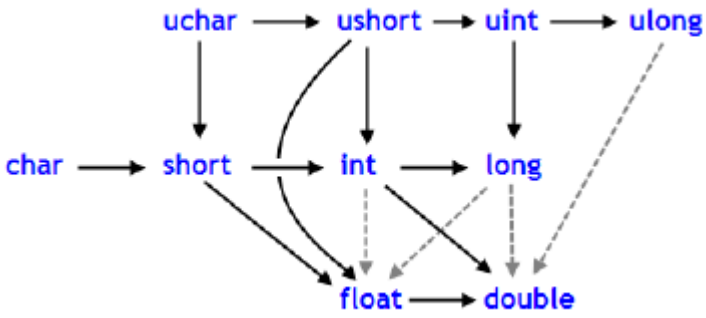
Method matrix/vector	Analogous method in NumPy	Description
matrix matrix.CumSum(const int axis)		
double vector.Prod(const double initial=1) double matrix.Prod(const double initial=1) vector matrix.Prod(const int axis,const double initial=1)	prod	Restituisce il prodotto degli elementi matrice/vettore che può essere eseguito anche per l'asse dato
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	Modifica la forma di una matrice senza modificarne i dati
void matrix.Resize(const ulong rows,const ulong cols)	resize	Restituisce una nuova matrice con una forma e una dimensione modificate
bool matrix.SwapRows(const ulong row1, const ulong row2)		Scambia le righe in una matrice
bool matrix.SwapCols(const ulong col1, const ulong col2)		Scambia le colonne in una matrice
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Restituisce l'intervallo di valori di una matrice/vettore o dell'asse della matrice specificato, equivalente a Max() - Min()
double vector.Percentile(const int percent) double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)	percentile	Restituisce il percentile specificato di valori di elementi matrice/vettore o elementi lungo l'asse specificato. I valori validi del parametro 'percentuale' sono compresi nell'intervallo [0, 100]
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	Restituisce il quantile specificato di valori di elementi matrice/vettore o elementi lungo l'asse specificato. Il parametro 'percentuale' assume valori nell'intervallo [0, 1]
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	Calcola la mediana degli elementi matrice/vettore. La mediana è il valore medio che separa la metà più alta degli elementi array/vettore dalla metà più bassa degli elementi.
double vector.Average() double matrix.Average()	average	Calcola la media aritmetica dei valori di matrice/vettore. La somma dei pesi al denominatore non può

Method matrix/vector	Analogous method in NumPy	Description
vector matrix.Average(const int axis)		essere uguale a 0, ma alcuni pesi possono essere 0
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	Restituisce la deviazione standard dei valori degli elementi matrice/vettorie o degli elementi lungo l'asse dato
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	Calcola la varianza dei valori degli elementi matrice/vettore
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	Calcola il coefficiente di correlazione di Pearson (coefficiente di correlazione lineare). Il coefficiente di correlazione è compreso nell'intervallo [-1, 1]
vector vector.Correlate(const vector& v,enum mode)	correlate	Calcola la correlazione incrociata di due vettori. Il parametro 'mode' determina la modalità di calcolo della convoluzione lineare
vector vector.Convolve(const vector& v, enum mode)	convolve	Restituisce la convoluzione lineare e discreta di due vettori Il parametro 'mode' determina la modalità di calcolo della convoluzione lineare
matrix matrix.Cov() matrix vector.Cov(const vector& v); (resulting matrix 2 x 2)	cov	Calcola la matrice di covarianza. La covarianza di due campioni (due variabili casuali) è una misura della loro dipendenza lineare

Typecasting

Casting di Tipi Numerici

Spesso si verifica la necessità di convertire un tipo numerico in un altro. Non tutti i tipi numerici possono essere convertiti in altri. Ecco lo schema di casting consentito:



Le linee continue con le frecce indicano le modifiche che vengono eseguite quasi senza alcuna perdita di informazioni. Al posto del tipo char, può essere usato il tipo bool (entrambi prendono 1 byte di memoria), al posto del tipo int, può essere utilizzato il tipo color (4 byte), al posto del tipo long, può essere utilizzato il tipo datetime (prende 8 byte). Le quattro linee tratteggiate grigie, anch'esse a freccia, denotano conversioni, quando la perdita di precisione può verificarsi. Ad esempio, il numero di cifre nel numero intero pari a 123456789 (int) è superiore al numero di cifre che possono essere rappresentate da un float.

```
   int n=123456789;
   float f=n;     // il contenuto di f è uguale a 1.234567892E8
   Print("n = ",n,"   f = ",f);
   // risulta n= 123456789    f= 123456792.00000
```

Un numero convertito in float ha lo stesso ordine, ma è meno accurato. Le conversioni, inversamente alle frecce nere, possono essere eseguite con eventuale perdita di dati. Conversioni tra char ed uchar, short ed ushort, int ed uint, long ed ulong (conversioni da entrambe le parti), possono portare alla perdita di dati.

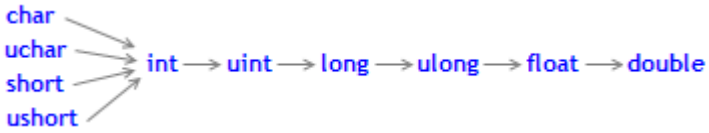
Come risultato di conversione di valori in virgola mobile verso il tipo intero, la parte frazionaria viene sempre eliminata. Se si desidera arrotondare un float al numero intero più vicino (che in molti casi è più utile), è necessario utilizzare MathRound ().

Esempio:

```
// --- Accelerazione gravitazionale
   double g=9.8;
   double round_g=(int)g;
   double math_round_g=MathRound(g);
   Print("round_g = ",round_g);
   Print("math_round_g = ",math_round_g);
/*
  Risultato:
   round_g = 9
   math_round_g = 10
```

```
*/
```

Se due valori vengono combinati da un operatore binario, prima dell'esecuzione dell'operazione l'operando di tipo inferiore viene convertito nel tipo superiore secondo la priorità data nel seguente schema:



I tipi di dati char, uchar, short, ed ushort incondizionatamente vengono convertiti nel tipo int.

Esempi:

```
   char   c1=3;
//--- Primo esempio
   double d2=c1/2+0.3;
   Print("c1/2 + 0.3 = ",d2);
// Risultato:   c1/2+0.3 = 1.3

//--- Secondo esempio
   d2=c1/2.0+0.3;
   Print("c1/2.0 + 0.3 = ",d2);
// Risultato:   c1/2.0+0.3 = 1.8
```

L'espressione calcolata si compone di due operazioni. Nel primo esempio, la variabile c1 del tipo char viene convertita in una variabile temporanea di tipo int, perché il secondo operando nell'operazione di divisione, la costante 2, è del tipo superiore int. Come risultato della divisione intera 3/2 si ottiene il valore 1, che è di tipo int.

Nella seconda operazione del primo esempio, il secondo operando è la costante 0.3, che è di tipo double, quindi il risultato della prima operazione viene convertito in una variabile temporanea di tipo double con il valore di 1.0 .

Nel secondo esempio la variabile di tipo char c1 viene convertita in una variabile temporanea di tipo double, perché il secondo operando nell'operazione di divisione, la costante 2.0, è di tipo double; non vengono effettuate altre conversioni.

Typecasting di Tipi Numerici

Nelle espressioni del linguaggio MQL5 entrambi il typecasting esplicito ed implicito, può essere utilizzato. Il typecasting esplicito è scritto come segue:

```
var_1 = (tipo)var_2;
```

Un risultato di un'espressione o di esecuzione di una funzione può essere utilizzato come variabile var_2. La registrazione funzionale del typecasting esplicito è anche possibile:

```
var_1 = tipo(var_2);
```

Consideriamo un typecasting esplicito sulla base del primo esempio.

```
//--- Terzo esempio
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2 + 0.3 = ",d2);
// Risultato:   (double)c1/2+0.3 = 1.80000000
```

Prima che l'operazione di divisione venga eseguita, la variabile c1 è esplicitamente 'castata'(_* neologismo, da cast/casting) al tipo double. Ora il numero costante integer 2 è castato al valore 2.0 valore di tipo double, perché come risultato della conversione del primo operando ha preso il tipo double. Infatti, il typecasting esplicito è un'operazione unaria.

Inoltre, quando si cerca di castare i tipi, il risultato può andare oltre il range permissibile. In questo caso, avviene un troncamento. Ad esempio:

```
   char c;
   uchar u;
   c=400;
   u=400;
   Print("c = ",c); // Risultato c=-112
   Print("u = ",u); // Risultato u=144
```

Prima che le operazioni (ad eccezione di quelle assegnazione) vengano eseguite, i dati vengono convertiti nel tipo massima priorità. Prima che vengano eseguite le operazioni di assegnazione, i dati vengono castati nel tipo target(_*di destinazione).

Esempi:

```
   int    i=1/2;         // nessun casting del tipo, il risultato è 0
   Print("i = 1/2  ",i);

   int k=1/2.0;          // l'espressione viene castata al tipo double,
   Print("k = 1/2  ",k);  // quindi al tipo int, ed il risultato è 0

   double d=1.0/2.0;     // nessun casting del tipo, il risultato è 0.5
   Print("d = 1/2.0; ",d);

   double e=1/2.0;       // l'espressione è castata al tipo double,
   Print("e = 1/2.0; ",e);// che è lo stesso del tipo target, il risultato è 0.5

   double x=1/2;         // l'espressione di tipo int viene castata al target di tipo d
   Print("x = 1/2; ",x);  // il risultato è 0.0
```

Durante la conversione di tipo long/ulong in double, la precisione può venir persa nel caso in cui il valore integer sia maggiore di 9223372036854774784 o minore di -9223372036854774784.

```
voidOnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- definisce il valore integer più alto, che non perde precisione quando viene cast
   while(l_max!=long((double)l_max))
      l_max--;
//--- definisce il valore integer più basso, che non perde precisione quando viene cas
```

```
   while(l_min!=long((double)l_min))
      l_min++;
//--- deriva l'intervallo trovato per valori interi
   PrintFormat("Quando si casta un valore integer a double, dev'essere"
               "all'interno dell'intervallo [%I64d, %I64d] ",l_min,l_max);
//--- ora, vediamo cosa succede se il valore cade al di fuori di questo intervallo
   PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
               l_max+1,double(l_max+1),long(double(l_max+1)));
   PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
               l_min-1,double(l_min-1),long(double(l_min-1)));
//--- visualizza il seguente risultato
// Il casting di un valore integer a double, dovrebbe essere entro l'intervallo [-9221
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_ma
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_
  }
```

Typecasting per il tipo String

Il tipo stringa ha la priorità più alta tra i tipi semplici. Pertanto, se uno degli operandi di un'operazione è di tipo stringa, il secondo operando sarà castato in una stringa automaticamente. Si noti che per una stringa, è possibile una singola operazione di addizione diadica a doppio-luogo. Il casting esplicito di stringa in un qualsiasi tipo numerico, è consentito.

Esempi:

```
   string s1=1.0/8;            // l'espressione viene castata al tipo double,
   Print("s1 = 1.0/8; ",s1);     //  quindi nel tipo target stringa,
// il risultato è "0.12500000" (una stringa contenente 10 caratteri)

   string s2=NULL;             // deinizializzazione stringa
   Print("s2 = NULL; ",s2);      // il risultato è una stringa vuota
   string s3="Ticket N"+12345; // l'espressione viene castata al tipo stringa
   Print("s3 = \"Ticket N\"+12345",s3);

   string str1="true";
   string str2="0,255,0";
   string str3="2009.06.01";
   string str4="1.2345e2";
   Print(bool(str1));
   Print(color(str2));
   Print(datetime(str3));
   Print(double(str4));
```

Typecasting di Puntatori di Classe Base a Puntatori di Classi Derivate

Oggetti di classi aperte generate possono anche essere visti come oggetti della classe base corrispondente. Questo porta ad alcune conseguenze interessanti. Per esempio, nonostante il fatto che gli oggetti di diverse classi, generati da una singola classe di base, possono differire significativamente l'uno dall'altro, possiamo creare una lista collegata (List) di essi, giacchè li vediamo come oggetti del tipo di base. Ma il contrario non è vero: gli oggetti della classe base non sono automaticamente gli oggetti di una classe derivata.

È possibile utilizzare il casting esplicito per convertire i puntatori della classe base in puntatori di una classe derivata. Ma si deve essere pienamente fiduciosi nella ammissibilità di tale trasformazione, perché altrimenti un errore runtime critico si verificherà e il programma MQL5 sarà interrotto.

Dynamic typecasting using dynamic_cast operator

Dynamic typecasting is performed using dynamic_cast operator that can be applied only to pointers to classes. Type validation is performed at runtime. This means that the compiler does not check the data type applied for typecasting when dynamic_cast operator is used. If a pointer is converted to a data type which is not the actual type of an object, the result is NULL.

```
dynamic_cast <type-id> ( expression )
```

The *type-id* parameter in angle brackets should point to a previously defined class type. Unlike C++, *expression* operand type can be of any value except for void.

Example:

```
class CBar { };
class CFoo : public CBar { };

void OnStart()
  {
   CBar bar;
//--- dynamic casting of *bar pointer type to *foo pointer is allowed
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // no critical error
   Print(foo);                             // foo=NULL
//--- an attempt to explicitly cast a Bar type object reference to a Foo type object
   foo=(CFoo *)&bar;                       // critical runtime error
   Print(foo);                             // this string is not executed
  }
```

Vedi anche

Tipi di Dati

Tipo Void e Costante NULL

Sintatticamente il tipo void è un tipo fondamentale insieme ai tipi char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double e string. Questo tipo è utilizzato per indicare che la funzione non restituisce alcun valore, o come parametro di funzione che indica l'assenza di parametri.

La variabile costante predefinita **NULL** è di tipo *void*. Può essere assegnato a variabili di altri tipi fondamentali senza conversione. Il confronto tra variabili di tipo fondamentale con il valore **NULL** è consentito.

Esempio:

```
//--- Se la stringa non viene inizializzata, allora assegniamo il nostro valore predei
if(some_string==NULL) some_string="empty";
```

Anche, **NULL**, può essere paragonato a puntatori a oggetti creati con l'operatore new.

Vedi anche

Variabili, Funzioni

I tipi definiti dall'utente

La parola chiave typedef in C++ permette di creare tipi di dati definiti dall'utente. Per farlo, è sufficiente specificare un nuovo nome di tipo di dati per un tipo di dati già esistenti. Il nuovo tipo di dati non viene creato. Un nuovo nome per il tipo esistente viene definito, invece. I tipi definiti dall'utente rendono le applicazioni più flessibili: a volte, è sufficiente cambiare istruzioni typedef utilizzando macro sostitutive (#define). I tipi definiti dall'utente anche migliorano la leggibilità del codice dal momento che è possibile applicare nomi personalizzati ai tipi di dati standard utilizzando typedef. Il formato generale della voce per la creazione di un tipo definito dall'utente:

```
typedef type new_name;
```

Qui, *type* significa qualsiasi tipo di dati accettabile, mentre *new_name* è un nuovo nome del tipo. Un nuovo nome viene impostato soltanto in aggiunta (non in sostituzione) a un nome di tipo esistente. MQL5 permette di creare puntatori a funzioni utilizzando typedef.

Puntatore alla funzione

Un puntatore a una funzione è generalmente definito nel seguente formato

```
typedef tipo_risultato_funzione*Nome_tipo_funzione)(lista_di_tipi_di_parametri_di_
```

dove dopo typedef, viene impostata la firma di funzione (numero e tipo dei parametri di input, così come un tipo di risultato restituito dalla funzione). Ecco un semplice esempio di creazione ed applicazione di un puntatore ad una funzione:

```
//--- dichiarare un puntatore a una funzione che accetta due parametri int
   typedef int (*TFunc)(int,int);
//--- TFunc è un tipo, ed è possibile dichiarare la variabile puntatore alla funzione
   TFunc func_ptr; // puntatore alla funzione
//--- dichiara le funzioni corrispondenti alla descrizione TFunc
   int sub(int x,int y) { return(x-y); }  // sottrae un numero da un altro
   int add(int x,int y) { return(x+y); }  // aggiunta di due numeri
   int neg(int x)       { return(~x);  }  // inverte i bit nella variabile
//--- la variabile func_ptr può memorizzare l'indirizzo della funzione da dichiarare s
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // errore: neg non ha il tipo int (int,int)
   Print(func_ptr(10));    // errore: mancano due parametri
```

In questo esempio, la variabile *func_ptr* può ricevere le funzioni *sub* e *add* poiché hanno due input ciascuno di tipo int definito nel puntatore alla funzione *TFunc*. Al contrario, la funzione *neg* non può essere assegnata al puntatore func_ptr giacchè dalla sua firma è diversa.

Organizzazione di modelli di eventi nell'interfaccia utente

Puntatori a funzioni permettono di creare facilmente l'elaborazione di eventi durante la creazione di un'interfaccia utente. Facciamo un esempio, dalla sezione CButton per mostrare come creare bottoni

ed aggiungere le funzioni per il loro handling. In primo luogo, definire un puntatore alla funzione *TAction* da chiamare premendo il bottone e creare tre funzioni secondo la *descrizione TAction*.

```
//--- crea un tipo di funzione personalizzata
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Apre il file                                                    |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s funzione chiamata (nome=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Salva il file                                                   |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s funzione chiamata (nome=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Chiude il file                                                  |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s funzione chiamata (nome=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

Quindi, crea la classe MyButton da CButton, dove dovremmo aggiungere il puntatore alla funzione *TAction*.

```
//+------------------------------------------------------------------+
//| Creare la classe bottone con la funzione di elaborazione eventi  |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // chart events handler
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- costruttore specificando il testo del bottone e il puntatore all' event handl
                     MyButton(string text, TAction act)
     {
      Text(text);
      m_action=act;
     }
```

```
   //--- imposta la funzione personalizzata chiamata dall'event handler OnEvent()
   void              SetAction(TAction act){m_action=act;}
   //--- standard chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- chiama l'handler custom m_action()
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- restituisce la chiamate dell'handler dalla classe genitore CButton
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
```

Crea la classe derivata CControlsDialog da[CAppDialog](), aggiunge l'array *m_buttons* ad essal per memorizzare i pulsanti tel tipo *MyButton*, nonché i metodi *AddButton(MyButton &button)* e *CreateButtons()* .

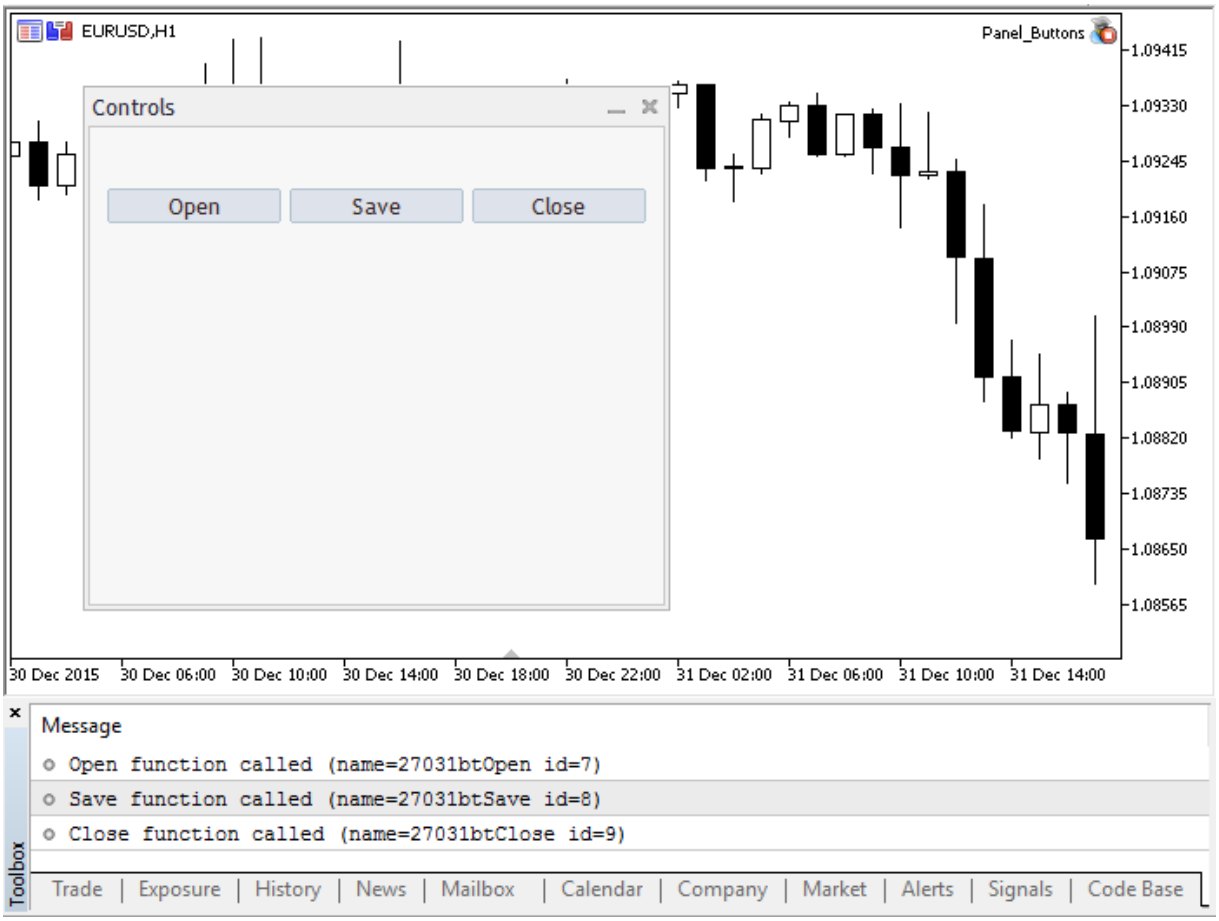
```
//+------------------------------------------------------------------+
//| CControlsDialog class                                            |
//| Obiettivo: pannello grafico per la gestione dell'applicazione    |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // array del bottone
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- crea
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- aggiunge il bottone
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- crea i bottoni
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Crea l'oggetto CControlsDialog sul chart                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
```

```
  }
//+------------------------------------------------------------------+
//| i defines                                                        |
//+------------------------------------------------------------------+
//--- indents e gaps
#define INDENT_LEFT                         (11)      // indentazione da sinistra (co
#define INDENT_TOP                          (11)      // indentazione dall'alto(con pe
#define CONTROLS_GAP_X                      (5)       // gap da coordinata X
#define CONTROLS_GAP_Y                      (5)       // gap da coordinata Y
//--- per i bottoni
#define BUTTON_WIDTH                        (100)     // grandezza da coordinate X
#define BUTTON_HEIGHT                       (20)      // grandezza da coordinata Y
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // grandezza da coordinata Y
//+------------------------------------------------------------------+
//| Creare ed aggiunge bottoni al pannello CControlsDialog           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- calcula le coordinate dei buttoni
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- aggiunge oggetti bottoni con puntatori a funzioni
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- creare i bottoni graficamente
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Fallimento nel creare il bottone %s %d",b.Text(),i);
         return(false);
        }
      //--- aggiungere ogni bottone al contenitore CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- successo
   return(true);
  }
```

Ora, siamo in grado di sviluppare il programma utilizzando il pannello di controllo CControlsDialog avente 3 pulsanti: Apri, Salva e Chiudi. Quando si fa clic su un bottone, la funzione appropriata nella forma di puntatore *TAction* viene chiamata.

```
//--- dichiara l'oggetto a livello globale per creare automaticamente quando si lancia
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ora, crea l'oggetto sul chart
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- avvia l'applicazione
   MyDialog.Run();
//--- applicazione inizializzata con successo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- distrugge la finestra
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Funzione di evento chart dell' Expert                            |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // ID evento
                  const long& lparam,   // parametro evento del tipo long
                  const double& dparam, // parametro event del tipo double
                  const string& sparam) // paraemetro evento del tipo string
  {
//--- chiama l'handler della classe padre (qui è CAppDialog) per gli eventi chart
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

L'apparenza dell'applicazione lanciata e il click dei bottoni viene fornita sulla schermata.



Il codice sorgente completo del programma

```
//+------------------------------------------------------------------+
//|                                                Panel_Buttons.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+

#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Il pannello con vari bottoni CButton"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| i defines                                                        |
//+------------------------------------------------------------------+
//--- indents e gaps
#define INDENT_LEFT                         (11)      // indentazione da sinistra (cor
#define INDENT_TOP                          (11)      // indentazione dall'alto(con pe
#define CONTROLS_GAP_X                      (5)       // gap da coordinata X
#define CONTROLS_GAP_Y                      (5)       // gap da coordinata Y
```

```
//--- per i bottoni
#define BUTTON_WIDTH                       (100)      // grandezza da coordinate X
#define BUTTON_HEIGHT                      (20)       // grandezza da coordinata Y
//--- for the indication area
#define EDIT_HEIGHT                        (20)       // grandezza da coordinata Y

//--- crea il tipo della funzione custom
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Apre il file                                                    |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s funzione chiamata (nome=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Salva il file                                                   |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s funzione chiamata (nome=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Chiude il file                                                  |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s funzione chiamata (nome=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| Creare la classe bottone con la funzione di elaborazione eventi  |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // chart events handler
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- costruttore specificando il testo del bottone e il puntatore all' event handl
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- imposta la funzione personalizzata chiamata dall'event handler OnEvent()
```

```
   void              SetAction(TAction act){m_action=act;}
   //--- standard chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- chiama l'handler custom
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- restituisce la chiamate dell'handler dalla classe genitore CButton
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| CControlsDialog class                                            |
//| Obiettivo: pannello grafico per la gestione dell'applicazione    |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // array del bottone
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- crea
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- aggiunge il bottone
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- crea i bottoni
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Crea l'oggetto CControlsDialog sul chart                         |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| Creare ed aggiunge bottoni al pannello CControlsDialog           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
```

```
//--- calcula le coordinate dei buttoni
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- aggiunge oggetti bottoni con puntatori a funzioni
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- creare i bottoni graficamente
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Fallimento nel creare il bottone %s %d",b.Text(),i);
         return(false);
        }
      //--- aggiungere ogni bottone al contenitore CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- successo
   return(true);
  }
//--- dichiara l'oggetto a livello globale per creare automaticamente quando si lancia
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ora, crea l'oggetto sul chart
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- avvia l'applicazione
   MyDialog.Run();
//--- applicazione inizializzata con successo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- distrugge la finestra
   MyDialog.Destroy(reason);
```

```
  }
//+------------------------------------------------------------------+
//| Funzione di evento chart dell' Expert                            |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // ID evento
                  const long& lparam,   // parametro evento del tipo long
                  const double& dparam, // parametro event del tipo double
                  const string& sparam) // paraemetro evento del tipo string
  {
//--- chiama l'handler della classe padre (qui è CAppDialog) per gli eventi chart
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Guarda anche

variabili, Funzioni

Puntatori agli Oggetti

MQL5 consente la creazione dinamica di oggetti di tipo complesso. Questo viene fatto utilizzando l'operatore 'new' che restituisce un descrittore dell'oggetto creato. La dimensione del descrittore è 8 byte. Sintatticamente, i descrittori di oggetti in MQL5 sono simili ai puntatori C++.

Esempio:

```
MyObject* hobject= new MyObject();
```

A differenza di C++, la variabile "hobject" dell'esempio sopra non è un puntatore alla memoria, ma è un descrittore di oggetti. Inoltre, in MQL5, tutti gli oggetti nei parametri della funzione devono essere passati per riferimento. Gli esempi seguenti mostrano il passaggio di oggetti come parametri di funzione:

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
 //--- costruttori e distruttori
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
 //--- inizializzo l'oggetto Foo
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
//+------------------------------------------------------------------+
//| Funzione start del programma script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiaro l'oggetto come una variabile, con creazione automatica
   Foo foo1;
//--- variante del passaggio di un oggetto per riferimento
   PrintObject(foo1);

//--- dichiaro un puntatore a un oggetto e lo creo usando l'operatore 'new'
   Foo *foo2=new Foo("foo2");
//--- variante del passaggio di un puntatore ad un oggetto per riferimento
   PrintObject(foo2); // il puntatore all'oggetto viene convertito automaticamente da

//--- dichiaro un array di oggetti Foo
```

```
   Foo foo_objects[5];
//--- variante del passaggio di un array di oggetti
   PrintObjectsArray(foo_objects); // una funzione separata per passare un array di og

//--- dichiaro un array di puntatori a oggetti di tipo Foo
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- variante del passaggio di un array di puntatori
   PrintPointersArray(foo_pointers); // una funzione separata per passare un array di

//--- prima di finire, assicurati di eliminare gli oggetti creati come puntatori
   delete(foo2);
//--- rimuovi l'array di puntatori
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
// Gli oggetti sono passati sempre per riferimento                   |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Object name=",object.m_name);
  }
//+------------------------------------------------------------------+
// Passaggio di un array di oggetti                                  |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
// Passaggio di un array di puntatori di oggetti                     |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
```

Controllare il puntatore prima dell'uso

Un tentativo di accedere a un puntatore non valido provoca l'arresto critico del programma. La funzioneCheckPointer viene utilizzata per controllare un puntatore prima dell'uso. Il puntatore può non essere valido nei seguenti casi:

- il puntatore è uguale a NULL;
- l'oggetto è stato distrutto utilizzando l'operatore delete.

Questa funzione può essere utilizzata per convalidare un puntatore. Un valore diverso da zero indica che è possibile accedere ai dati su questo puntatore.

```
class CMyObject
 {
protected:
  double              m_value;
public:
                      CMyObject(void);
                      CMyObject(double value) {m_value=value;};
                     ~CMyObject(void){};
  //---
  double              Value(void) {return(m_value);}
 };
//+------------------------------------------------------------------+
//| Funzione start del programma script                              |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- creo un oggetto non inizializzato
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- inizializzo il puntatore
  pointer=new CMyObject(M_PI);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- elimino l'oggetto
  delete(pointer);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
```

```
   3. pointer is POINTER_INVALID
*/
```

Per convalidare rapidamente il puntatore, puoi anche utilizzare l'operatore "!" (LNOT) che lo verifica tramite una chiamata implicita della funzione CheckPointer. Ciò consente una scrittura del codice più concisa e chiara. Di seguito sono riportati i controlli dell'esempio precedente:

```
//+------------------------------------------------------------------+
//| Funzione start del programma script                              |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- creo un oggetto non inizializzato
  CMyObject *pointer;
  if(!pointer)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- inizializzo il puntatore
  pointer=new CMyObject(M_PI);
  if(!pointer)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- elimino l'oggetto
  delete(pointer);
  if(!pointer)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. pointer is POINTER_INVALID
*/
```

L'operatore "==" viene utilizzato per un rapido controllo di NULL. Per esempio: ptr==NULL or ptr!=NULL.

Vedi anche

Variabili, Inizializzazione delle Variabili, Visibilità Campo e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Riferimenti: Modificatore & e Parola Chiave this

Passaggio di parametri per Riferimento

In MQL5 parametri di tipo semplice possono essere passati sia per valore che per riferimento, mentre i parametri di tipi compositi vengono sempre passati per riferimento. Per informare il compilatore che un parametro deve essere passato per riferimento, il carattere 'e commerciale' & viene aggiunto prima del nome del parametro.

Il passaggio di un parametro per riferimento significa passare l'indirizzo della variabile, è per questo che tutte le variazioni del parametro che viene passato per riferimento verranno immediatamente riflesse nella variabile sorgente. Utilizzando il passaggio di parametro per riferimento, è possibile implementare il ritorno dei svariati risultati di una funzione allo stesso tempo. Al fine di impedire la variazione di un parametro passato per riferimento, utilizzare il modificatore const.

Pertanto, se il parametro di input di una funzione è un array, un oggetto di struttura o di classe, il simbolo '&' viene posto nell'intestazione della funzione dopo il tipo di variabile e prima del nome.

Esempio

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
  };
//+------------------------------------------------------------------------------+
//| riempie l'array                                                              |
//+------------------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
     ArrayResize(m_array,ArraySize(array));
     ArrayCopy(m_array, array);
     }
  }
```

Nell'esempio sopra la classe viene dichiarata la CDemoClass, la qualeche contiene il membro privato - array m_array[] di tipo double. Funzione setArray() viene dichiarata, alla quale array[] è passato per riferimento. Se l'intestazione funzione non contiene l'indicazione di passaggio per riferimento, cioè non contiene il carattere 'e commerciale', un messaggio di errore viene generato al tentativo di compilare tale codice.

Nonostante il fatto che la matrice viene passata per riferimento, non possiamo assegnare un array ad un altro. Abbiamo bisogno di effettuare la copia elemento-per-elemento dei contenuti della matrice di origine alla matrice destinataria. La presenza dei & nella descrizione della funzione è la condizione obbligatoria per array e strutture quando viene passata come parametro della funzione.

Parola chiave this

Una variabile di tipo classe (oggetto) può essere passata sia per riferimento e per puntatore. Così come per il riferimento, il puntatore permette di avere accesso ad un oggetto. Dopo che il puntatore oggetto viene dichiarato, il nuovo operatore deve essere applicato ad esso per crearlo ed inizializzarlo.

La parola riservata **this** è designata per ottenere il riferimento dell'oggetto stesso, che è disponibile all'interno di metodi di classe o struttura. **this** sempre fa riferimento all'oggetto, nel metodo di cui è utilizzato, e l'espressione GetPointer(this) dà il puntatore dell'oggetto, il cui componente è la funzione, in cui viene eseguita la chiamata di GetPointer(). In MQL5 funzioni non possono restituire oggetti, ma possono restituire il puntatore dell'oggetto.

Quindi, se abbiamo bisogno che una funzione restituisca un oggetto, siamo in grado di restituire il puntatore di questo oggetto sottoforma di GetPointer(this). Aggiungiamo la funzione getDemoClass() che restituisce il puntatore del oggetto di questa classe, nella descrizione di CDemoClass.

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------------------------+
//| riempie l'array                                                                    |
//+------------------------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------------------------+
//| restituisce il suo puntatore                                                       |
//+------------------------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

Le strutture non hanno puntatori, gli operatori *new* e *delete* non possono essere applicati ad essi, GetPointer(this) non può essere utilizzato.

Vedi anche

Puntatori Oggetto, Creazione ed Eliminazione di Oggetti, Visibilità Campo e Durata di Variabili

Operazioni ed Espressioni

Alcuni caratteri e sequenze di caratteri sono di particolare importanza. Questi sono i cosiddetti simboli delle operazioni, ad esempio:

```
+ - * / %     Simboli di operazioni aritmetiche
&& ||         Simboli di operazioni logiche
= += *=       Caratteri degli operatori di assegnazione
```

Simboli delle operazioni vengono utilizzati nelle espressioni ed hanno senso quando vengono dati ad essi operandi appropriati. I segni di punteggiatura sono enfatizzati, anche. Si tratta di parentesi, virgola, due punti e punto e virgola.

Simboli di operazioni, segni di punteggiatura e spazi vengono utilizzati per separare gli elementi del linguaggio gli uni dagli altri.

Questa sezione contiene la descrizione dei seguenti argomenti:

- Expressions
- Arithmetical Operations
- Operazioni di assegnazione
- Operations of Relation
- Boolean Operations
- Operazioni bit per bit
- Altre Operazioni
- Priorità ed Ordine delle operazioni

Expressions

Un'espressione è composta da uno o più operandi e simboli di funzionamento. Un'espressione può essere scritta in più linee.

Esempi:

```
a++; b = 10;          // in una riga ci sono varie espressioni
//---  un'espressione è divisa in varie righe
x = (y * z) /
    (w + 2) + 127;
```

Un'espressione che termina con un punto e virgola (;) è un operatore.

Vedi anche

Regole di Precedenza

Operazioni Aritmetiche

Operazioni aritmetiche includono operazioni di addizione e moltiplicazione:

```
Somma delle variabili                        i = j + 2;
Differenza delle variabili                   i = j - 3;
Modifica del segno                           x = -x;
Prodotto delle variabili                     z = 3 * x;
Divisione quoziente                          i = j / 5;
Il resto della divisione                     minutes = time % 60;
Aggiunge 1 al valore della variabile         i++;
Aggiunge 1 al valore della variabile         ++i;
Sottrae 1 dal valore della variabile         k--;
Sottrae 1 dal valore della variabile         --k;
```

Le operazioni di incremento e decremento sono applicate solo alle variabili, non possono essere applicate alle costanti. I prefissi incremento (++i) e decremento (--k) vengono applicati a destra variabile, giusto prima che questa variabile venga utilizzata in un'espressione.

Post-incremento (i++) e post-decremento (k--) vengono applicati alla variabile, giusto dopo che questa vnga utilizzata in un'espressione.

Avviso importante

```
int i=5;
int k = i++ + ++i;
```

Problemi computazionali possono verificarsi durante lo spostamento della suddetta espressione da un ambiente di programmazione ad un altro (per esempio, da Borland C++ ad MQL5). In generale, l'ordine dei calcoli dipende dall'implementazione del compilatore. In pratica, ci sono due modi per implementare il post-decremento (e post-incremento):

1. Il post-decremento (ed il post-incremento) viene applicato alla variabile dopo il calcolo dell'intera espressione.
2. Il post-decremento (post-incremento) viene applicato alla variabile immediatamente all'operazione.

Attualmente per i calcoli è implementato in MQL5 il primo metodo di post-decremento (e di post-incremento). Ma anche conoscendo questa sua peculiarità, non è consigliabile sperimentare con il suo uso.

Esempi:

```
int a=3;
a++;             // espressione valida
int b=(a++)*3;   // espressione non valida
```

Vedi anche

Regole di Precedenza

Operazioni di assegnazione

Il valore dell'espressione che include l'operazione data è il valore dell'operando sinistro dopo l'assegnazione:

```
Assegnazione del valore x alla variabile y                    y = x;
```

Le seguenti operazioni uniscono operazioni aritmetiche o di bit con operazioni di assegnazione:

```
Aggiungere x alla variabile y                                 y += x;
Sottrarre x dalla y variabile y                               y -= x;
Moltiplicare la variabile y per x                             y *= x;
Dividere la variabile y per x                                 y /= x;
Resto della divisione della variabile y per x                 y %= x;
Slittamento della rappresentazione binaria di y a destra di x bits    y >>= x;
Slittamento della rappresentazione binaria di y a sinistra di x bit   y <<= x;
Operazione AND di bit della rappresentazione binaria di y ed x        y &= x;
Operazione OR di bit della rappresentazione binaria di y ed x         y |= x;
Operazione OR Esclusivo di bit della rappresentazione binaria di y ed x  y ^= x;
```

Operazioni di bit possono essere applicate solo a numeri interi. Quando si esegue l'operazione di slittamento logico della rappresentazione y verso destra/sinistra di x-bit, vengono utilizzate le 5 cifre binarie più piccole del valore x, le più alte vengono scartate, cioè il passaggio è fatto per 0-31 bit.

Con l'operazione %= (valore y dal modulo di x), il segno del risultato è uguale al segno del numero diviso.

L'operatore di assegnazione può essere utilizzato più volte in un'espressione. In questo caso la trasformazione dell'espressione viene eseguita da sinistra a destra:

```
y=x=3;
```

In primo luogo, alla variabile x viene assegnato il valore 3, poi alla variabile y viene assegnato il valore di x, vale a dire 3, anche.

Vedi anche

Regole di Precedenza

Operations of Relation

Il booleano FALSE (_*falso) è rappresentato da un valore intero pari a zero, mentre il booleano TRUE (_*vero) è rappresentato da qualsiasi valore diverso da zero.

Il valore delle espressioni che contengono operazioni di relazione oppure operazioni logiche è FALSE(0) o TRUE(1).

```
True se a è uguale a b                    a == b;
True se a non è uguale a b                a != b;
True se a è minore di b                   a < b;
True se a è maggiore di b                 a > b;
True se a è minore o uguale a b           a <= b;
True se a è maggiore o uguale a b         a >= b;
```

L'uguaglianza di due numeri reali non può essere confrontata. Nella maggior parte dei casi, due numeri apparentemente identici possono essere diversi a causa di diversi valori nel 15esimo posto decimale. Per poter confrontare correttamente due numeri reali, confrontare la differenza normalizzata di questi numeri con lo zero.

Esempio:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
voidOnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%.16f e %.16f sono uguali",first,third);
     }
  }
// Result: 0.3000000000000000  0.2999999999999998   sono uguali
```

Vedi anche

Regole di Precedenza

Boolean Operations

Negazione Logica NOT (!)

L'operando della negazione logica (!) deve essere di tipo aritmetico. Il risultato è TRUE (1), se il valore operando è FALSE (0), ed è uguale a FALSE (0), se l'operando è diverso da FALSE (0).

```
if(!a) Print("not 'a'");
```

Operazione Logica OR (||)

Operazione Logica OR (||) dei valori x e y. Il valore dell'espressione è TRUE (1), se almeno uno dei due tra il valore di x oppure il valore y sono veri (cioè non nulli). In caso contrario - FALSE (0).

```
if(x<0 || x>=max_bars) Print("fuori dal range");
```

Operazione Logica AND (&&)

Operazione Logica AND (&&) dei valori x e y. Il valore dell'espressione è TRUE (1), se il valore di x ed il valore di y sono entrambi veri (cioè non nulli). In caso contrario - FALSE (0).

Stima sintetica delle Operazioni Booleane

Lo schema della cosiddetta "stima breve" viene applicato ad operazioni booleane, ossia il calcolo dell'espressione è terminato quando il risultato dell'espressione può essere stimato con precisione.

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- il primo esempio della stima breve
   if(func_false() && func_true())
     {
      Print("Operazione &&: Non vedrai mai questa espressione");
     }
   else
     {
      Print("Operazione &&: Il risultato della prima espressione è falso, così la secc
     }
//--- il secondo esempio della stima breve
   if(!func_false() || !func_true())
     {
      Print("Operazione ||: Il risultato della prima espressione è vero, così la secor
     }
   else
     {
      Print("Operazione ||: Non vedrai mai questa espressione");
     }
  }
//+------------------------------------------------------------------------------+
```

```
//| La funzione restituisce sempre falso                             |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("Function func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| La funzione restituisce sempre vero                              |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("Function func_true()");
   return(true);
  }
```

Vedi anche

Regole di Precedenza

Operazioni bit per bit

Complemento ad Uno

Complemento del valore della variabile fino ad uno. Il valore dell'espressione contiene 1 in tutte le cifre dove il valore della variabile contiene 0, e 0 in tutte le cifre dove la variabile contiene 1.

```
b = ~n;
```

Esempio:

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// Il risultato sarà:
// a = 97   b = -98
```

Slittamento a Destra

La rappresentazione binaria di x è spostata a destra di y cifre. Se il valore per slittamento è del tipo senza segno, viene fatto lo slittamento logico destra, cioè i bit liberati del lato sinistro saranno riempiti con zeri.

Se il valore per lo slittamento è di tipo segnato, viene fatto lo slittamento aritmetico a destra, cioè le cifre liberate del lato-sinistro verranno riempitecon il valore di un bit di segno (se il numero è positivo, il valore del bit di segno è 0 , se il numero è negativo, il valore del bit di segno è 1).

```
x = x >> y;
```

Esempio:

```
   char a='a',b='b';
   Print("Prima:  a = ",a, "  b = ",b);
//--- slittamento verso destra
   b=a>>1;
   Print("Dopo:   a = ",a, "  b = ",b);
// Il risultato sarà:
// Prima:  a = 97   b = 98
// Dopo:   a = 97   b = 48
```

Slittamento sinistro

La rappresentazione binaria di x viene slittata verso sinistra di y cifre, i numeri liberati sul lato-destro vengono riempiti con zeri.

```
x = x << y;
```

Esempio:

```
   char a='a',b='b';
   Print("Prima:  a = ",a, "  b = ",b);
//--- slittamento a sinistra
   b=a<<1;
```

```
   Print("Dopo:   a = ",a, "  b = ",b);
// Il risultato sarà:
// Prima:  a = 97   b = 98
// Prima:   a = 97   b = -62
```

Non è consigliabile slittare per un numero di bit maggiore o uguale alla lunghezza della variabile spostata, perché il risultato di tale operazione è indefinito.

L'operazione AND di bit

L'operazione AND di bit, di rappresentazioni x ed y codificate in binario. Il valore dell'espressione contiene un 1 (VERO) in le cifre dove x ed y contengono non-zeri, e contiene 0 (FALSO) in tutte le altre cifre.

```
b = ((x & y) != 0);
```

Esempio:

```
   char a='a',b='b';
//--- Operazione AND
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// Il risultato sarà:
// a = 97   b = 98
// a & b = 96
```

L'operazione OR di bit

L'operazione OR di bit di rappresentazioni binarie di x e y. Il valore dell'espressione contiene 1 in tutte le cifre dove x o y non contengono 0, e contiene 0 in tutte le altre cifre.

```
b = x | y;
```

Esempio:

```
   char a='a',b='b';
//--- Operazione OR
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// Il risultato sarà:
// a = 97   b = 98
// a | b = 99
```

L' operazione OR Esclusivo di bit

L'operazione OR Esclusivo di bit (eXclusive OR) di rappresentazioni binarie di x e y. Il valore dell'espressione contiene un 1 in tutte le cifre dove x e y hanno valori binari diversi, e contiene 0 in tutte le altre cifre.

```
b = x ^ y;
```

Esempio:

```
   char a='a', b='b';
//--- Operazione di Esclusione OR
   char c=a^b;
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// Il risultato sarà:
// a = 97   b = 98
// a ^ b = 3
```

Operazioni di bit vengono eseguite solo con interi.

Vedi anche

Regole di Precedenza

Altre Operazioni

Indicizzazione ([])

Quando si indirizza l' i-esimo elemento dell'array, il valore dell' espressione è il valore di una variabile con il numero di serie i.

Esempio:

```
array[i] = 3; // Assegna il valore di 3 all' i-esimo elemento dell'array.
```

Solo un numero intero può essere indice di un array. Array a quattro dimensioni ed inferiori, sono ammessi. Ogni misura è indicizzata da 0 a **misurazione della grandezza**-1. Nel caso particolare, per un array unidimensionale costituito da 50 elementi, il riferimento al primo elemento sarà simile ad array [0], e l'ultimo elemento sarà array [49].

Quando si indirizza oltre l'array, il sottosistema di esecuzione genererà un errore critico, ed il programma viene interrotto.

Chiamare Funzione con x1, x2,..., xn Argomenti

Ogni argomento può rappresentare una costante, una variabile o un'espressione del tipo corrispondente. Gli argomenti passati vengono separati da virgole e devono essere all'interno di parentesi, la parentesi di apertura deve seguire il nome della funzione chiamata.

Il valore dell' espressione è il valore restituito dalla funzione. Se il valore restituito è di tipo void, tale chiamata di funzione non può essere posizionata a destra nell' operazione di assegnazione. Si noti che le espressioni x1,...,xn vengono eseguite esattamente in questo ordine.

Esempio:

```
   int length=1000000;
   string a="a",b="b",c;
// --- Altre Operazioni
   int start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("tempo per 'c = a + b' = ",(stop-start)," millisecondi, i = ",i);
```

Operazione virgola (,)

Espressioni separate da virgole vengono eseguite da sinistra a destra. Tutti gli effetti del calcolo dell' espressione di sinistra possono apparire prima che l'espressione di destra sia calcolata. Il tipo di risultato e valore coincidono con quelli dell'espressione di destra. L'elenco dei parametri da passare (vedi sopra) può essere considerato come un esempio.

Esempio:

```
for(i=0,j=99; i<100; i++,j--) Print(array[i][j]);
```

Operatore punto (.)

Per l' accesso diretto ai membri pubblici delle strutture e classi viene utilizzato l'operatore punto. Sintassi:

```
Nome_della_variabile_del_tipo_struttura.Nome_del_membroe
```

Esempio:

```
   struct SessionTime
     {
      string sessionName;
      int    startHour;
      int    startMinutes;
      int    endHour;
      int    endMinutes;
     } st;
   st.sessionName="Asian";
   st.startHour=0;
   st.startMinutes=0;
   st.endHour=9;
   st.endMinutes=0;
```

Operazione Risoluzione Ambito (::)

Ogni funzione in un programma MQL5 ha il proprio ambito di esecuzione. Per esempio, la funzione di sistema Print() viene eseguita in un ambito globale. Funzioni Importate vengono chiamate nell'ambito dell'importazione corrispondente. Metodi di funzioni classi hanno l'ambito delle classi corrispondenti. La sintassi dell'operazione di risoluzione dell'ambito è la seguente:

```
[Scope_name]::Function_name(parameters)
```

Se non c'è un nome ambito, questa è direzione esplicita di utilizzare l'ambito globale. Se non viene eseguita alcuna operazione di risoluzione dell'ambito, la funzione è ricercata nel più vicino ambito. Se non esiste una funzione in ambito locale, la ricerca viene eseguita in ambito globale.

L'operazione di risoluzione dell'ambito viene utilizzata anche per definire la funzionemembro della classe.

```
type Class_name::Function_name(parameters_description)
   {
// corpo della funzione
   }
```

L'uso di diverse funzioni con lo stesso nome di diversi contesti di esecuzione in un programma può causare ambiguità. L'ordine di priorità delle chiamate di funzione senza esplica specificazione dello scopo è la seguente:

1. I metodi della classe. Se nessuna funzione con il nome specificato viene impostata nella classe, passare al livello successivo.

2. Funzioni MQL5. Se il linguaggio non dispone di una tale funzione, passare al livello successivo.
3. Funzioni globali definite dall'utente. Se non viene trovata alcuna funzione con il nome specificato, passare al livello successivo.
4. Funzioni importate. Se non viene trovata alcuna funzione con il nome specificato, il compilatore restituisce un errore.

Per evitare l'ambiguità di chiamate di funzione, sempre specificare esplicitamente l'ambito funzione utilizzando l'operazione di risoluzione della visibilità/scopo(scope).

Esempio:

```
#property script_show_inputs
#import "kernel32.dll"
   int GetLastError(void);
#import

class CCheckContext
  {
   int          m_id;
public:
               CCheckContext() { m_id=1234; }
protected:
   int          GetLastError() { return(m_id); }
  };
class CCheckContext2 : public CCheckContext
  {
   int          m_id2;
public:
               CCheckContext2() { m_id2=5678; }
   void         Print();
protected:
   int          GetLastError() { return(m_id2); }
  };
void CCheckContext2::Print()
  {
   ::Print("Terminal GetLastError",::GetLastError());
   ::Print("kernel32 GetLastError",kernel32::GetLastError());
   ::Print("parent GetLastError",CCheckContext::GetLastError());
   ::Print("our GetLastError",GetLastError());
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   CCheckContext2 test;
   test.Print();
  }
```

Operazione di Ottenimento della Grandezza del Tipo di Dato o Grandezza di qualsiasi Tipo di Dato di Oggetto (sizeof)

Utilizzando l'operazione sizeof, può essere definita la grandezza di memoria corrispondente ad un identificatore di tipo. L'operazione sizeof è del seguente formato:

Esempio:

```
sizeof(espressione)
```

Qualsiasi identificatore, o nome del tipo racchiuso tra parentesi può essere utilizzato come espressione. Si noti che il nome di tipo void non può essere utilizzato, e l'identificatore non può appartenere al campo dei bit, o essere un nome di funzione.

Se l'espressione è il nome di un array statico (cioè è data la prima dimensione), allora il risultato è la grandezza dell'intero array (cioè il prodotto del numero di elementi e la lunghezza del tipo). Se l'espressione è il nome di un array dinamico (la prima dimensione non è specificata), il risultato sarà la grandezza dell'oggetto dell' array dinamico.

Quando sizeof viene applicato al nome del tipo struttura o classe, o all'identificatore del tipo struttura o classe, il risultato è la grandezza effettiva della struttura o classe.

Esempio:

```
struct myStruct
  {
   char   h;
   int    b;
   double f;
  } str;
Print("sizeof(str) = ",sizeof(str));
Print("sizeof(myStruct) = ",sizeof(myStruct));
```

La grandezza viene calcolata in fase di compilazione.

Vedi anche

Regole di Precedenza

Regole di Precedenza

Each group of operations in the table has the same priority. The higher the priority of operations is, the higher it is position of the group in the table. La regole di precedenza determinano il raggruppamento delle operazioni ed operandi.

Attenzione: La precedenza delle operazioni in linguaggio MQL5 corrisponde alla priorità adottato in C++, e si differenzia dalla priorità data nel linguaggio MQL4.

Operazione	Descrizione	Ordine di Esecuzione
() [] .	Chiamate di funzione Riferito ad un elemento dell'array Riferito ad un elemento della struttura	Da sinistra a destra
! ~ - ++ -- (type) sizeof	Negazione logica Negazione di bit (complemento) Cambio di segno Incremento di uno Decremento di uno Typecasting Determinazione della grandezza in byte	Da destra a sinistra
* / %	Moltiplicazione Divisione Modulo divisione	Da sinistra a destra
+ -	Addizione Sottrazione	Da sinistra a destra
<< >>	Spostamento a sinistra Spostamento a destra	Da sinistra a destra
< <= > >=	Minore di Minore o uguale a Maggiore di Maggiore o uguale a	Da sinistra a destra
== !=	Uguale a Non uguale a	Da sinistra a destra
&	L'operazione AND di bit	Da sinistra a destra
^	L' operazione OR Esclusivo di bit	Da sinistra a destra
//| |	L'operazione OR di bit	Da sinistra a destra
&&	Operazione logica AND	Da sinistra a destra
||	Operazione logica OR	Da sinistra a destra
?:	Operatore condizionale	Da destra a sinistra
= *= /=	Assegnazione Moltiplicazione con assegnazione Divisione con assegnazione	Da destra a sinistra

Operazione	Descrizione	Ordine di Esecuzione
%= += -= <<= >>= &= ^= |=	Modulo con assegnazione Addizione con assegnazione Sottrazione con assegnazione Spostamento a sinistra con assegnazione Spostamento a destra con assegnazione AND di bit con assegnazione OR Esclusivo di bit con assegnazione OR di bit con assegnazione	
,	Virgola	Da sinistra a destra

Per modificare l'ordine di esecuzione dell'operazione, vengono usate le parentesi, che sono di più alta priorità.

Operatori

Gli operatori del linguaggio descrivono alcune operazioni algoritmiche che devono essere eseguite per realizzare un compito. Il corpo del programma è una sequenza di tali operatori. Gli operatori si susseguono uno ad uno, e sono separati da un punto e virgola.

Operatore	Descrizione
Operatore Composto {}	Uno o più operatori di qualsiasi tipo, racchiuso tra parentesi graffe {}
Espressione operatore (;)	Qualsiasi espressione che termina con un punto e virgola (;)
operatore return	Termina la corrente funzione e restituisce il controllo al programma chiamante
if-else operatore condizionale	Viene utilizzato quando è necessario fare una scelta
?: operatore condizionale	Un analogo semplice dell'operatore condizionale if-else
operatore di selezione switch	Passa il controllo all'operatore, che corrisponde al valore dell' espressione
operatore ciclico while	Esegue un operatore fino a quando l'espressione controllata diventa falsa. L'espressione viene controllata prima di ogni iterazione
operatore ciclico for	Esegue un operatore fino a quando l'espressione controllata diventa falsa. L'espressione viene controllata prima di ogni iterazione
operatore ciclico do-while	Esegue un operatore fino a quando l'espressione controllata diventa falsa. La condizione di fine è verificata, dopo ogni ciclo. Il corpo del ciclo viene sempre eseguito almeno una volta.
operatore break	Termina l'esecuzione del più vicino operatore switch, while, do-while o for, collegato
operatore continue	Passa il controllo all'inizio del più vicino operatore ciclo esterno, while, do-while o for
operatore new	Crea un oggetto di grandezza appropriata e restituisce un descrittore dell'oggetto creato.
operatore delete	Elimina l'oggetto creato dall'operatore new

Un operatore può occupare una o più righe. Due o più operatori possono trovarsi nella stessa riga. Gli operatori che controllano oltre l'ordine di esecuzione (if, if-else, switch, while e for) possono essere annidati l'uno nell'altro.

Esempio:

```
if(Month() == 12)
  if(Day() == 31) Print("Felice Anno Nuovo!");
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Composto

Un operatore composto (un blocco) è costituito da uno o più operatori di qualsiasi tipo, racchiusi tra parentesi graffe {}. La parentesi graffa di chiusura non deve essere seguita da un punto e virgola (;).

Esempio:

```
if(x==0)
  {
   Print("invalid position x = ",x);
   return;
  }
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Espressione

Qualsiasi espressione seguita da un punto e virgola (;) è un operatore. Ecco alcuni esempi di operatori di espressione.

Operatore di assegnazione

Identificatore = espressione;

```
x=3;
y=x=3;
bool equal=(x==y);
```

L' operatore di assegnazione può essere utilizzato più volte in un'espressione. In this case, the expression is processed from right to left.

Function Calling Operator

Nome_Funzione (argomento1,..., argomentoN);

```
FileClose(file);
```

Operatore Vuoto

Costituito solo da un punto e virgola (;) e viene usato per indicare un corpo vuoto di un operatore di controllo.

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Return

L'operatore return termina l'esecuione della corrente funzione e restituisce il controllo al programma chiamante. Il risultato del calcolo dell'espressione viene restituito alla funzione chiamante. L'espressione può contenere un operatore di assegnazione.

Esempio:

```
int CalcSum(int x, int y)
  {
   return(x+y);
  }
```

In funzioni con il tipo di ritorno void, l' opratore return deve essere utilizzato senza espressione:

```
void SomeFunction()
  {
   Print("Ciao!");
   return;     // questo operatore può essere rimosso
  }
```

La parentesi graffa destra della funzione significa esecuzione implicita dell' operatore return senza l'espressione.

Che cosa può essere restituito: tipi semplici, strutture semplici, puntatori agli oggetti. Con l' operatore *return* non si è in grado di restituire nessun array, oggetti della classe, variabili di tipo composto struttura.

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

If-Else Operatore Condizionale

L'operatore IF-ELSE viene utilizzato quando deve essere fatta una scelta. Formalmente, la sintassi è la seguente:

```
if (espressione)
    operatore1
else
    operatore2
```

Se l'espressione è vera, operatore1 viene eseguito ed il controllo viene dato all'operatore che segue operatore2 (operatore2 non viene eseguito). Se l'espressione è falsa, operatore2 viene eseguito.

La parte else dell'operatore if può essere omessa. Così, una divergenza può apparire in operatori if annidati con la parte else omessa. In questo caso, else indirizza al più vicino precedente operatore if nello stesso blocco che non ha la parte else.

Esempi:

```
//--- La parte else si riferisce alla seconda se l'operatore:
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- La parte else si riferisce al primo se l'operatore:
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- Operatori nidificati
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERROR");
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Ternario ?:

La forma generale dell' operatore ternario è la seguente:

```
espressione1 ? espressione2 : espressione3
```

Per il primo operando - "expression1" - qualsiasi espressione che si traduce in valore di tipo bool può essere utilizzata. Se il risultato è true, allora viene eseguito l'operatore stabilito dal secondo operando, vale a dire "expression2".

Se il primo operando è false, viene eseguito il terzo operando - "expression3". Il secondo operando ed il terzo operando, vale a dire "expression2" ed "expression3" devono restituire i valori di un certo tipo e non devono essere di tipo void. Il risultato dell'esecuzione dell' operatore condizionale è il risultato di expression2 o risultato dei expression3, a seconda del risultato di expression1.

```
//--- normalizzare la differenza tra il prezzo di apertura e chiusura per un interval
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Questa voce è equivalente alla seguente:

```
   double true_range;
   if(High==Low)true_range=0;                // se High e Low sono uguali
   else true_range=(Close-Open)/(High-Low); // se il range non è nullo
```

Restrizioni uso operatori

Sulla base del valore di "expression1", l'operatore deve restituire uno dei due valori - o "expression2" o "expression3". Ci sono numerose limitazioni a queste espressioni:

1. Non confondere il tipo definito dall'utente con il tipo semplice o l' enumerazione. NULL può essere utilizzato per puntatore.
2. Se tipi di valori sono semplici, l'operatore sarà del tipo massimo (vedere Tipo di casting).
3. Se uno dei valori è un' enumerazione ed il secondo è di tipo numerico, l'enumerazione viene sostituita da int e la seconda regola viene applicata.
4. Se entrambi i valori sono enumerazioni, i loro tipi devono essere identici, e l'operatore sarà di tipo enumerazione.

Restrizioni per i tipi definiti-dall-utente (classi o strutture):

a) I tipi devono essere identici o uno deve essere derivato derivato dall'altro.

b) Se i tipi non sono identici (ereditarietà), allora il figlio è implicitamente castato al genitore, cioè l'operatore sarà di tipo genitore.

c) Da non confondere l'oggetto ed il puntatore - entrambe le espressioni sono o oggetti o puntatori. NULL non può essere usato per il puntatore.

Nota

Fare attenzione quando si usa l'operatore condizionale come argomento di una funzione overloaded, perchè il tipo di risultato di un operatore condizionale è definito al momento della compilazione del programma. And this type is defined as the larger of the types "expression2" and "expression3".

Esempio:

```
void func(double d) { Print("argomento double: ",d); }
void func(string s) { Print("argomento string: ",s); }

bool   Expression1=true;
double Expression2=M_PI;
string Expression3="3.1415926";

voidOnStart()
  {
   func(Expression2);
   func(Expression3);

   func(Expression1?Expression2:Expression3);   // attenzione al casting implito a sti
   func(!Expression1?Expression2:Expression3);  // attenzione al casting implicito a s
  }

//   Risultato:
//   argomento double: 3.141592653589793
//   argomento string: 3.1415926
//   argomento string: 3.141592653589793
//   argomento string: 3.1415926
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Switch

Confronta il valore di espressione con costanti in tutte le varianti *case* e passa il controllo all'operatore che corrisponde al valore dell' espressione. Ogni variante di *case* può essere contrassegnata con una costante intera, una costante letterale o un'espressione costante. L'espressione costante non può contenere variabili o chiamate di funzione. L' espressione dell'operatore *switch* deve essere di tipo integer - int oppure uint.

```
switch(expression)
  {
   case costante: operatori
   case costante: operatori
      ...
   default: oeratori
  }
```

Gli operatori contrassegnati dall'etichetta *default* vengono eseguiti se nessuna delle costanti nell'operatore *case* è uguale al valore dell'espressione. La variante *default* non deve essere necessariamente dichiarata e non deve essere necessariamente l'ultima. Se nessuna delle costanti corrisponde al valore dell' espressione e la variante *default* non è disponibile, non vengono eseguite azioni.

La parola chiave *case* con una costante sono solo etichette, e se gli operatori vengono eseguiti per una certa variante *case*, il programma esegue ulteriormente gli operatori di tutte le varianti successive fino a che non si verifica l'operatore *break*. Permette di collegare una sequenza di operatori con diverse varianti.

Un'espressione costante è calcolata in fase di compilazione. Non ci sono due costanti in un operatore *switch* che possono avere lo stesso valore.

Esempi:

```
//--- Primo esempio
switch(x)
  {
   case 'A':
      Print("CASE A");
      break;
   case 'B':
   case 'C':
      Print("CASE B or C");
      break;
   default:
      Print("NOT A, B or C");
      break;
  }

//---  Secondo esempio
   string res="";
   int i=0;
   switch(i)
```

```
      {
       case 1:
          res=i;break;
       default:
          res="default";break;
       case 2:
          res=i;break;
       case 3:
          res=i;break;
      }
   Print(res);
/*
   Result
   default
*/
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Cicico While

L'operatore while consiste di un operatore di espressione controllata e l'operatore che deve essere soddisfatto:

```
while(espressione)
 operatore;
```

Se l'espressione è vera, l'operatore viene eseguito finchè l'espressione non diventa falsa. Se l'espressione è falsa, il controllo viene passato al prossimo operatore. Il valore dell'espressione è definito prima che l'operatore venga eseguito. Pertanto, se l'espressione è falsa dall'inizio, l'operatore non verrà eseguito per niente.

Nota

Se si prevede che in un ciclo saranno trattate un gran numero di iterazioni, è consigliabile che si vada a verificare il fatto della terminazione forzata del programma utilizzando la funzioneIsStopped().

Esempio:

```
while(k<n && !IsStopped())
  {
   y=y*x;
   k++;
  }
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Ciclico For

L'operatore si compone di tre espressioni ed un operatore eseguibile:

```
for(espressione1; espressione2; espressione3)
  operatore;
```

Espressione1 descrive l'inizializzazione del ciclo. Espressione2 verifica le condizioni della terminazione del ciclo. Se è vera, il corpo del ciclo for viene eseguito. Il ciclo si ripete fino a quando expression2 diventa falsa. Se è false, il ciclo viene terminato, ed il controllo viene dato all'operatore successivo. Espressione3 viene calcolata dopo ogni iterazione.

L' operatore for è equivalente alla seguente successione di operatori:

```
espressione1;
while(Espressione2)
  {
  operatore;
  espressione3
  };
```

Una qualunque delle tre, o tutte e tre, le espressioni, possono essere assenti nell'operatore for, ma il punto e virgola (;) che le separa non deve essere omesso. Se espressione2 è omessa, è considerata sempre vero. L'operatore for(;;) è un loop continuo, equivalente all'operatore while(1). Ogni espressione 1 o 3 può essere costituita da diverse espressioni combinate tramite un operatore virgola ','.

Nota

Se si prevede che in un ciclo saranno trattate un gran numero di iterazioni, è consigliabile che si vada a verificare il fatto della terminazione forzata del programma utilizzando la funzioneIsStopped().

Esempi:

```
for(x=1;x<=7000; x++)
  {
   if(IsStopped())
      break;
   Print(MathPower(x,2));
  }
//--- Altro esempio
for(;!IsStopped();)
  {
   Print(MathPower(x,2));
   x++;
   if(x>10) break;
  }
//--- Terzo esempio
for(i=0,j=n-1;i<n && !IsStopped();i++,j--) a[i]=a[j];
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Ciclico do while

I cicli for e while verificano la terminazione all'inizio, non alla fine di un ciclo. Il terzo operatore ciclico do - while verifica la condizione di terminazione alla fine, dopo ogni iterazione del ciclo. Il corpo del ciclo viene sempre eseguito almeno una volta.

```
do
  operatore;
while(espressione);
```

Prima l'operatore viene eseguito, poi l'espressione viene calcolata. Se è vero, allora l'operatore viene eseguito nuovamente, e così via. Se l'espressione diventa falsa, il ciclo termina.

Nota

Se si prevede che in un ciclo saranno trattate un gran numero di iterazioni, è consigliabile che si vada a verificare il fatto della terminazione forzata del programma utilizzando la funzioneIsStopped().

Esempio:

```
//--- Calcolo della serie di Fibonacci
   int counterFibonacci=15;
   int i=0,first=0,second=1;
   int currentFibonacciNumber;
   do
     {
      currentFibonacciNumber=first+second;
      Print("i = ",i,"  currentFibonacciNumber = ",currentFibonacciNumber);
      first=second;
      second=currentFibonacciNumber;
      i++; // senza questo operatore apparirà un loop infinito!
     }
   while(i<counterFibonacci && !IsStopped());
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Break Operator

L'operatore break interrompe l'esecuzione del più vicino esteriore, nidificato: switch, while, do-while o for operatore. Il controllo viene passato all'operatore che segue quello terminato. Uno degli scopi di questo operatore è terminare il ciclo di esecuzione quando un certo valore viene assegnato ad una variabile.

Esempio:

```
//--- ricerca per il primo elemento zero
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Continue

L' operatore continue passa il controllo all'inizio dell'operatore di ciclo esterno più vicino while, do-while o for, all'iterazione successiva chiamata. Lo scopo di questo operatore è opposto a quello dell' operatore break.

Esempio:

```
//--- Somma di tutti gli elementi diversi da zero
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
     }
   return(sum);
  }
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore new di Creazione Oggetto

L'operatore new crea automaticamente un oggetto di una grandezza corrispondente, chiama il costruttore dell'oggetto e restituisce un descrittore di oggetto creato. In caso di fallimento, l'operatore restituisce un descrittore null che può essere confrontato con la costante NULL.

Il nuovo operatore può essere applicato solo ad oggetti classe. Non può essere applicato a strutture.

L'operatore non deve essere usato per creare array di oggetti. Per effettuare questa operazione, utilizzare la funzione ArrayResize().

Esempio:

```
//+------------------------------------------------------------------------------+
//| Creazione figura                                                             |
//+------------------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- a caso crea una delle 7 forme possibili
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- draw
   if(m_shape!=NULL)
     {
      //--- pre-settings
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- draw
      m_shape.Draw();
     }
//---
  }
```

Va notato che il descrittore oggetto non è un puntatore all' indirizzo di memoria.

Un oggetto creato con l'operatore new deve essere esplicitamente rimosso usando l'operatore delete.

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Operatore Cancellatore Oggetto delete

L'operatore delete elimina un oggetto creato dall'operatore new, chiama il distruttore di classe corrispondente e libera la memoria occupata dall'oggetto. Un descrittore reale di un oggetto esistente viene utilizzato come operando. Dopo che l'operazione di eliminazione viene eseguita, il descrittore dell'oggetto non è più valido.

Esempio:

```
      //--- elimina figura
      delete m_shape;
      m_shape=NULL;
      //--- crea una nuova figura
      NewShape();
```

Vedi anche

Inizializzazione delle Variabili, Visibilità Ambito e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Funzioni

Ogni operazione può essere divisa in sottoattività, ciascuna delle quali può essere direttamente rappresentata sotto forma di un codice, o suddivisa in piccole sotto-attività. Questo metodo è chiamato *raffinamento per passi successivi*. Le funzioni vengono utilizzate per scrivere il codice di sotto-compiti da risolvere. Il codice che descrive ciò che è una funzione, viene chiamato *definizione di funzione:*

```
intestazione_funzione
  {
  le istruzioni
  }
```

Tutto ciò che è prima della prima parentesi graffa è l' *intestazione* della definizione di funzione, e ciò che è tra le parentesi graffe è il *corpo* della definizione della funzione. L'intestazione funzione comprende una descrizione del tipo di valore di ritorno, il nome (identificatore) e parametri formali. Il numero di parametri passati alla funzione è limitato e non può superare i 64.

La funzione può essere chiamata da altre parti del programma, tante volte quanto necessario. Infatti, il tipo di ritorno, identificatore della funzione e tipi di parametri costituiscono la *funzione prototipo*.

Il prototipo della funzione è la dichiarazione della funzione, ma non la sua definizione. A causa della dichiarazione esplicita del tipo di return e l' elenco dei tipi di argomento, un controllo di tipo restrittivo ed il typecasting implicito, sono possibili durante le chiamate di funzione. Molto spesso le dichiarazioni di funzione vengono utilizzate nelle classi per migliorare la leggibilità del codice.

La definizione della funzione deve corrispondere esattamente alla sua dichiarazione. Ogni funzione dichiarata deve essere definita.

Esempio:

```
double                          // tipo di valore di ritorno(_*return)
linfunc (double a, double b) // nome funzione ed elenco parametri
  {
                                // valore di ritorno
   return (a + b);              // dell'operatore composito
  }
```

L'operatore return può restituire il valore di un'espressione che trova in questo operatore. Se necessario, il valore dell' espressione viene convertito nel tipo di risultato della funzione. Cosa può essere restituito: tipi semplici, strutture semplici, puntatori agli oggetti. Con l'operatore *return* non puoi restituire nessun array, oggetti classe, variabili di tipo struttura composta.

Una funzione non restituisce un valore dev'essere descritta come tipo void.

Esempio:

```
void errmesg(string s)
  {
   Print("errore: "+s);
  }
```

I parametri passati alla funzione possono avere valori di default, che sono definiti dalle costanti di quel tipo.

Esempio:

```
int unaqualchefunzione(double a,
              double d=0.0001,
              int n=5,
              bool b=true,
              string s="stringa passata")
  {
   Print("Parametro Richiesto a = ",a);
   Print("Passa i seguenti parametri: d = ",d," n = ",n," b = ",b," s = ",s);
   return(0);
  }
```

Se uno dei parametri ha un valore di default, tutti i parametri successivi devono avere anche valori di default.

Esempio di dichiarazione inesatta:

```
int unaqualchefunzione(double a,
              double d=0.0001,     // valore di default 0.0001 dichiarato
              int n,               // il valore di default non specificato !
              bool b,              // il valore di default non specificato !
              string s="stringa passata")
  {
  }
```

Vedi anche

Overload, Funzioni Virtuali, Polimorfismo

Chiamate di funzione

Se un nome che non è stato descritto in precedenza, appare nell'espressione ed è seguito da parentesi sinistra, viene contestualmente considerato come il nome di una funzione.

```
nome_funzione (x1, x2,..., xn)
```

Argomenti (parametri formali) vengono passati per valore, ad es. ogni espressione x1,..., xn viene calcolata, ed il valore viene passato alla funzione. L'ordine di calcolo delle espressioni e l'ordine di caricamento dei valori non sono garantiti. Durante l'esecuzione, il sistema controlla il numero e il tipo di argomenti passati alla funzione. Tal modo di affrontare la funzione viene chiamato una chiamata valore.

Chiamata di funzione è un'espressione, il cui valore è il valore restituito dalla funzione. Il tipo di funzione sopra descritto deve corrispondere con il tipo del valore di ritorno. La funzione può essere dichiarata o descritta in qualsiasi parte del programma a portata globale, cioè, al di fuori di altre funzioni. La funzione non può essere dichiarata o descritta all'interno di un'altra funzione.

Esempi:

```
int start()
  {
   double some_array[4]={0.3, 1.4, 2.5, 3.6};
   double a=linfunc(some_array, 10.5, 8);
   //...
  }
double linfunc(double x[], double a, double b)
  {
   return (a*x[0] + b);
  }
```

Alla chiamata di una funzione con parametri di default, l'elenco dei parametri da passare può essere limitato, ma non prima del primo parametro predefinito.

Esempi:

```
void somefunc(double init,
              double sec=0.0001, //impostazione valori default
              int level=10);
//...
somefunc();                       // Chiamata sbagliata. Il primo parametro deve essere
somefunc(3.14);                   // Chiamata corretta
somefunc(3.14,0.0002);            // Chiamata corretta
somefunc(3.14,0.0002,10);         // Chiamata corretta
```

Quando si chiama una funzione, non si possono saltare parametri, anche quelli con valori di default:

```
somefunc(3.14, , 10);           / / Chiamata sbagliata -> il secondo parametro è stato
```

Vedi anche

Overload, Funzioni Virtuali, Polimorfismo

Passaggio di parametri

Ci sono due metodi, con cui il linguaggio macchina può passare argomenti ad un sottoprogramma (funzione). Il primo metodo è quello di inviare un parametro per valore. Questo metodo copia il valore dell' argomento in un parametro di funzione formale. Pertanto, le variazioni di questo parametro all'interno della funzione non hanno alcuna influenza sull'argomento della corrispondente chiamata.

```
//+------------------------------------------------------------------------------+
//| Passaggio di parametri per valore                                            |
//+------------------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   int a=14,b=8;
   Print("a e b prima della chiamata:",a," ",b);
   double d=FirstMethod(a,b);
   Print("a e b dopo la chiamata:",a," ",b);
  }
//--- Risultato dell'esecuzione dello script
// a e b prima della chiamata: 14 8
// a e b dopo la chiamata: 14 8
```

Il secondo metodo è passare per riferimento. In questo caso, il riferimento ad un parametro (non al suo valore) viene passato ad un parametro di funzione. All'interno della funzione, viene utilizzato per fare riferimento al parametro attuale specificato nella chiamata. Ciò significa che le modifiche dei parametri influenzeranno l'argomento utilizzato per chiamare la funzione.

```
//+------------------------------------------------------------------------------+
//| Passaggio di parametri per riferimento                                       |
//+------------------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
```

```
//---
   return(res);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   int a=14,b=8;
   Print("a e b prima della chiamata:",a," ",b);
   double d=SecondMethod(a,b);
   Print("a e b dopo la chiamata:",a," ",b);
  }
//+------------------------------------------------------------------------------+
//--- risultato dell'esecuzione dello script
// a e b prima della chiamata: 14 8
// a e b dopo la chiamata: 28 4
```

MQL5 utilizza entrambi i metodi, con una sola eccezione: gli array, le variabili di tipo struttura e gli oggetti della classe sono sempre passati per riferimento. Al fine di evitare variazioni di parametri attuali (argomenti passati alla chiamata di funzione) utilizziamo lo specificatore di accesso const. Quando si tenta di modificare il contenuto di una variabile dichiarata con l'identificatore *const* , il compilatore genera un errore.

Nota

Va notato che i parametri vengono passati ad una funzione in ordine inverso, cioè, prima viene calcolato e passato l'ultimo parametro, e poi il penultimo, e così via. L'ultimo parametro calcolato e passato è quello che sta prima dopo la parentesi di apertura.

Esempio:

```
voidOnStart()
  {
//---
   int a[]={0,1,2};
   int i=0;

   func(a[i],a[i++],"First call (i = "+string(i)+")");
   func(a[i++],a[i],"Second call (i = "+string(i)+")");
//Risultato:
// First call (i = 0) : par1 = 1     par2 = 0
// Second call (i = 1) : par1 = 1     par2 = 1

  }
//+------------------------------------------------------------------------------+
//|                                                                              |
//+------------------------------------------------------------------------------+
void func(int par1,int par2,string comment)
```

```
  {
   Print(comment,": par1 = ",par1,"    par2 = ",par2);
  }
```

Nella prima chiamata (vedi esempio sopra) la variabile ***i*** viene prima usata nella concatenazione stringhe:

```
  "First call (i = "+string(i)+")"
```

Qui il suo valore non cambia. Poi la variabile ***i*** viene utilizzata nel calcolo dell'elemento dell' array ***a[i++]***, cioè quando si accede all'elemento dell'array con indice i, la variabile ***i*** viene incrementata. E solo dopo ciò, il primo parametro con il valore modificato di ***i*** viene calcolato.

Nella seconda chiamata lo stesso valore di i (calcolato sulla prima fase della chiamata di funzione) viene utilizzato per calcolare tutti i tre i parametri. Solo dopo che il primo parametro viene calcolato, la variabile ***i*** viene modificata di nuovo.

Vedi anche

Visibilità Ambito e Durata delle Variabili,L'Overload, Funzioni Virtuali, Polimorfismo

L'overloading di funzioni

Di solito il nome della funzione tende a riflettere il suo scopo principale. Di regola, i programmi leggibili e contengono vari ben selezionati identificatori. Talvolta funzioni diverse vengono utilizzate per gli stessi scopi. Consideriamo, per esempio, una funzione che calcola il valore medio di un array di numeri a doppia precisione e la stessa funzione, ma operante con un array di interi. Entrambi sono comodi da poter essere chiamati AverageFromArray:

```
//+------------------------------------------------------------------------------------+
//| Il calcolo della media per un array di tipo double                                 |
//+------------------------------------------------------------------------------------+
double AverageFromArray(const double & array[],int size)
  {
   if(size<=0) return 0.0;
   double sum=0.0;
   double aver;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // Sommatoria per il double
     }
   aver=sum/size;        // Divisione della somma per il numero
//---
   Print("Calcolo della media per un array di tipo double");
   return aver;
  }
//+------------------------------------------------------------------------------------+
/ / | Il calcolo della media per un array di tipo int                                  |
//+------------------------------------------------------------------------------------+
double AverageFromArray(const int & array[],int size)
  {
   if(size<=0) return 0.0;
   double aver=0.0;
   int sum=0;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];      // Sommatoria per l'int
     }
   aver=(double)sum/size;// Dare la quantità di tipo double, e dividere
//---
   Print("Calcolo della media per un array di tipo int");
   return aver;
  }
```

Ogni funzione contiene il messaggio output tramite la funzione Print();

```
   Print("Calcolo della media per un array di tipo int");
```

Il compilatore seleziona una funzione necessaria in conformità con i tipi degli argomenti e la loro quantità. La regola, secondo cui la scelta è fatta, è chiamata *algoritmo signature matching*. Una signature è un elenco dei tipi utilizzati per la dichiarazione della funzione.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   int    a[5]={1,2,3,4,5};
   double b[5]={1.1,2.2,3.3,4.4,5.5};
   double int_aver=AverageFromArray(a,5);
   double double_aver=AverageFromArray(b,5);
   Print("int_aver = ",int_aver,"   double_aver = ",double_aver);
  }
//--- Risultato dello script
// Calcola la media per un array di tipo int
// Calcola la media per un array di tipo double
// int_aver= 3.00000000    double_aver= 3.30000000
```

Il function overloading è un processo di creazione di più funzioni con lo stesso nome, ma parametri diversi. Ciò significa che in varianti overload di una funzione, il numero di argomenti e/o i loro tipi devono essere diversi. Una specifica variante di funzione viene selezionata in base alla corrispondenza della lista di argomenti quando si chiama la funzione, per l'elenco dei parametri nella dichiarazione di funzione.

Quando una funzione in overload viene chiamata, il compilatore deve avere un algoritmo per selezionare la funzione appropriata. L'algoritmo che esegue questa scelta dipende dai castings di quali tipi sono presenti. La migliore corrispondenza deve essere univoca. Una funzione overloadata deve essere il miglior connubio tra tutte le altre varianti di almeno un argomento. Allo stesso tempo, deve corrispondere per tutti gli altri argomenti, non peggio di altre varianti.

Di seguito è riportato un algoritmo di corrispondenza per ogni argomento.

Algoritmo di Scelta di una Funzione Overload

1. Utilizzare la corrispondenza stretta (se possibile).
2. Prova aumento tipo standard.
3. Prova typecasting standard.

L'incremento di tipo standard è migliore di altre conversioni standard. Incremento è la conversione di float to double, of bool, char, short or enum to int. Il Typecasting di array di simili tipi integer appartiene anche al typecasting. Tipi simili sono: bool, char, uchar, dal momento che tutti e tre i tipi sono interi a singolo-byte; interi a doppio-byte short ed ushort, interi 4-byte int, uint, e colour; long, ulong e datetime.

Naturalmente, lo strict matching è il migliore. Per ottenere una tale consistenza può essere utilizzato typecasting. Il compilatore non può far fronte a situazioni ambigue. Pertanto non si dovrebbe fare

affidamento su sottili differenze di tipi e conversioni implicite che rendono poco chiara la funzione di overload.

Se avete dei dubbi, utilizzare la conversione esplicita per garantire il rispetto rigoroso.

Esempi di funzioni in overloaded in MQL5 si possono vedere nell'esempio di funzioni ArrayInitialize().

Regole di funzioni overloading si applicano a overload dei metodi della classe.

L'overloading di funzioni del sistema è consentito, ma va osservato che il compilatore è in grado di selezionare accuratamente la funzione necessaria. Per esempio, siamo in grado di overloadare la funzione di sistema Mathmax() in 4 modi diversi, ma solo due varianti sono corrette.

Esempio:

```
// 1. overload consentito - la funzione differisce dalla funzione incorporata MathMax
double MathMax(double a,double b,double c);

// 2. l'overload non è permesso!
// il umero di parametri è diverso, ma l'ultimo ha un valore predefinito
// questo porta all'occultamento della funzione di sistema quando si chiama, che è ina
double MathMax(double a,double b,double c=DBL_MIN);

// 3. l'overload è consentito - l'overload normale per tipo di parametri a e b
double MathMax(int a,int b);

// 4. l'overload non è permesso!
// Il numero ed i tipi di parametri sono gli stessi come nell'originale double MathMax
int MathMax(double a,double b);
```

Vedi anche

Overload, Funzioni Virtuali, Polimorfismo

Operazione Overloading

Per facilità di lettura e scrittura di codice, l'overloading di alcune operazioni è consentito. L' operatore overloading è scritto usando la parola chiave operator. I seguenti operatori possono essere sottoposti ad overload:

- binari +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^
- unari +,-,++,--,!,~
- operatore di assegnazione =
- operatore di indicizzazione []

L'operazione di overloading consente l'utilizzo della notazione operativa (scritta in forma di espressioni semplici) per gli oggetti complessi - strutture e classi. La scrittura di espressioni utilizzando le operazioni di overload semplifica la visualizzazione del codice sorgente, perché una implementazione più complessa è nascosta.

Per esempio, consideriamo numeri complessi, che sono composti dalla parte reale e quella immaginaria. Essi sono ampiamente utilizzati in matematica. Il linguaggio MQL5 non ha un tipo di dati per rappresentare numeri complessi, ma è possibile creare un nuovo tipo di dati nella forma di una struttura o classe. Dichiarare la struttura complessa e definire i quattro metodi che implementano quattro operazioni aritmetiche:

```
//+------------------------------------------------------------------------------+
//| Una struttura per le operazioni con i numeri complessi                       |
//+------------------------------------------------------------------------------+
struct complex
  {
   double            re; // Parte reale
   double            im; // Parte immaginaria
   //--- Costruttori
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- Operazioni aritmetiche
   complex           Add(const complex &l,const complex &r) const;  // Adizione
   complex           Sub(const complex &l,const complex &r) const;  // Sottrazione
   complex           Mul(const complex &l,const complex &r) const;  // Moltiplicazione
   complex           Div(const complex &l,const complex &r) const;  // Divisione
  };
```

Ora, nel nostro codice possiamo dichiarare variabili che rappresentano i numeri complessi, e lavorare con esse.

Ad esempio:

```
voidOnStart()
  {
// --- Dichiara e inizializza le variabili del tipo complesso
   complex a(2,4),b(-4,-2);
```

```
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- Sommare due numeri
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- Moltiplicare due numeri
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Dividere due numeri
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

Ma sarebbe più conveniente utilizzare operatori consueti "+", "-", "*" e "/" per le operazioni aritmetiche ordinarie con numeri complessi.

L' operatore parola chiave viene utilizzato per la definizione di una funzione membro che esegue la conversione del tipo. Operazioni unarie e binarie per le variabili oggetto della classe possono essere sovraccaricate come funzioni membro non-statiche. Esse implicitamente agiscono sull' oggetto della classe.

Gran parte delle operazioni binarie possono essere sovraccaricate come funzioni regolari che accettano uno o entrambi gli argomenti come variabile della classe o un puntatore ad un oggetto di questa classe. Per il nostro tipo complesso, l'overloading nella dichiarazione sarà simile a questa:

```
   //--- Operatori
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
```

L'esempio completo dello script:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Dichiara ed inizializza le variabili di tipo complex
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
   //a.re=5;
   //a.im=1;
   //b.re=-1;
   //b.im=-5;
//--- Sommare due numeri
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
```

```
//--- Moltiplica i due numeri
   z=a*b;
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Dividere due numeri
   z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------------------+
//| Una struttura per le operazioni con i numeri complessi                       |
//+------------------------------------------------------------------------------+
struct complex
  {
   double            re; // Parte reale
   double            im; // Parte immaginaria
   //--- Costruttori
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- Operazioni aritmetiche
   complex           Add(const complex &l,const complex &r) const;  // Adizione
   complex           Sub(const complex &l,const complex &r) const;  // Sottrazione
   complex           Mul(const complex &l,const complex &r) const;  // Moltiplicazione
   complex           Div(const complex &l,const complex &r) const;  // Divisione
   //--- Operatori Binari
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
  };
//+------------------------------------------------------------------------------+
//| Addizione                                                                    |
//+------------------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- Risultato
   return res;
  }
//+------------------------------------------------------------------------------+
//| Sottrazione                                                                  |
//+------------------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
```

```
   complex res;
//---
   res.re=l.re-r.re;
   res.im=l.im-r.im;
//--- Risultato
   return res;
  }
//+------------------------------------------------------------------+
//| Moltiplicazione                                                  |
//+------------------------------------------------------------------+
complex complex::Mul(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re*r.re-l.im*r.im;
   res.im=l.re*r.im+l.im*r.re;
//--- Risultato
   return res;
  }
//+------------------------------------------------------------------+
//| Divisione                                                        |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- Numeri complessi vuoti
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- Controlla per lo zero
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": il numero è zero");
      return(res);
     }
//--- Variabili ausiliarie
   double e;
   double f;
//--- Seleziona variante di calcolo
   if(MathAbs(r.im)<MathAbs(r.re))
     {
      e = r.im/r.re;
      f = r.re+r.im*e;
      res.re=(l.re+l.im*e)/f;
      res.im=(l.im-l.re*e)/f;
     }
   else
     {
      e = r.re/r.im;
      f = r.im+r.re*e;
      res.re=(l.im+l.re*e)/f;
      res.im=(-l.re+l.im*e)/f;
```

```
     }
//--- Risultato
   return res;
  }
```

Gran parte delle operazioni unarie per le classi possono essere sovraccaricate come funzioni ordinarie che accettano un solo argomento di oggetto della classe o un puntatore ad essa. Aggiungere overloading di operazioni unarie "-" e "!".

```
//+------------------------------------------------------------------------------------+
//| Una struttura per le operazioni con i numeri complessi                             |
//+------------------------------------------------------------------------------------+
struct complex
  {
   double            re;       // Parte Reale
   double            im;       // Parte Immaginaria
...
   //--- Operatori Unari
   complex operator-()  const; // Meno unario
   bool    operator!()  const; // Negazione
  };
...
//+------------------------------------------------------------------------------------+
//| Fare l'Overloading dell'operatore "meno unario"                                    |
//+------------------------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- Risultato
   return res;
  }
//+------------------------------------------------------------------------------------+
//| Fare l'Overloading dell'operatore "negazione logica"                               |
//+------------------------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- Le parti reale ed immaginaria del numero complesso sono pari a zero?
   return (re!=0 && im!=0);
  }
```

Ora siamo in grado di controllare il valore di un numero complesso per lo zero ed ottenere un valore negativo:

```
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
```

```
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Dichiara ed inizializza le variabili di tipo complex
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- Divide i due numeri
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- Un numero complesso è uguale a zero per default (nel costruttore di default re==
  zero complesso;
   Print("!zero=",!zero);
//--- Assegna un valore negativo
   zero=-z;
   PrintFormat("z=%.2f+i*%.2f,  zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
   PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
//--- Controlla per lo zero ancora una volta
   Print("!zero=",!zero);
//---
  }
```

Si noti che non abbiamo dovuto overloadare l'operatore di assegnazione "=", come nelle strutture di tipi semplici che possono essere copiate direttamente una dentro l'altra. Quindi, possiamo ora scrivere un codice per i calcoli che coinvolgono numeri complessi nel modo consueto.

L'overloading dell'operatore indicizzazione permette di ottenere i valori degli array racchiusi in un oggetto, in modo semplice e familiare, e contribuisce anche ad una migliore leggibilità del codice sorgente. Per esempio, abbiamo bisogno di fornire l'accesso ad un simbolo nella stringa nella posizione specificata. Una stringa in MQL5 è un tipo separato stringa, che non è un array di simboli, ma con l'aiuto di un' operazione di indicizzazione sovraccarico si può fornire un lavoro semplice e trasparente nella classe CString generata:

```
//+------------------------------------------------------------------------------+
//| Classe per accedere ai simboli nella stringa come array di simboli           |
//+------------------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
void OnStart()
  {
//--- Un array per ricevere i simboli da una stringa
   int     x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
```

```
                  17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
   CString str("abcdefghijklmnopqrstuvwxyz[ ]CS");
   string  res;
//--- Fa una frase utilizzando i simboli dalla variabile str
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
      res+=ShortToString(str[x[i]]);
     }
//--- Mostra i risultati
   Print(res);
  }
```

Un altro esempio di sovraccarico della operazione di indicizzazione sono le operazioni con gli array. La matrice rappresenta un array bi-dimensionale dinamico, la dimensione della matrice non è definita in anticipo. Di conseguenza, non è possibile dichiarare un array di forma array [ ] [ ] senza specificare la dimensione della seconda dimensione, e quindi passare questo array come parametro. A possible solution is a special class CMatrix, which contains an array of CRow class objects.

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Operazioni di addizzione e moltiplicazione di matrici
   CMatrix A(3),B(3),C();
//--- Prepara un array per riga
   double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
   double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//--- Riempie la matrice
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- Da in output le matrici nel log degli Experts
   Print("---- Elementi della matrice A");
   Print(A.String());
   Print("---- Elementi della matrice B");
   Print(B.String());

//--- Addizione di matrici
   Print("---- Addizione di matrici A e B");
   C=A+B;
//--- Da in output la rappresentazione della stringa formattata
   Print(C.String());

//--- Moltiplicazione di matrici
   Print("---- Moltiplicazione delle matrici A e B");
   C=A*B;
   Print(C.String());

//--- Ora vi mostriamo come ottenere i valori nello stile di matrice array dinamica [
```

```
   Print("Da in output il valore della matrice C secondo gli elementi");
//--- Va attraverso le righe della matrice - oggetti CRow - in loop
   for(int i=0;i<3;i++)
     {
      string com="| ";
      //--- Forma righe dalla matrice per i valori
      for(int j=0;j<3;j++)
        {
         //--- Ottiene gli elementi della matrice dai numeri di righe e colonne
         double element=C[i][j];// [i] - Accesso a CRow nell'array m_rows[] ,
                                // [j] - Operatore overloaded, di indicizzazione in CR
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
      com+="|";
      //--- Da in output i valori di row
      Print(com);
     }
  }
//+------------------------------------------------------------------------------+
//| Classe "Row"                                                                 |
//+------------------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- Costruttori ed un distruttore
                     CRow(void)           { ArrayResize(m_array,0);     }
                     CRow(const CRow &r)  { this=r;                     }
                     CRow(const double &array[]);
                    ~CRow(void){};
   //--- Numero di elementi nella riga
   int               Size(void) const     { return(ArraySize(m_array));}
   //--- Restituisce la stringa con valori
   string            String(void) const;
   //--- Indicizzazione operatore
   double            operator[](int i) const  { return(m_array[i]);    }
   //--- Assegnazione operatori
   void              operator=(const double  &array[]); // Un array
   void              operator=(const CRow & r);         // Un altro oggetto CRow
   double            operator*(const CRow &o);          // Oggetto CRow per la moltip
  };
//+------------------------------------------------------------------------------+
//| Costruttore per inizializzare la riga con un array                           |
//+------------------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- Se l' array non è vuoto
```

```
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- Riempie con i valori
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
     }
//---
  }
//+------------------------------------------------------------------------------------+
//| Operatore di assegnazione per l'array                                              |
//+------------------------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- Riempie l'array con i valori
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------------------------+
//| Operatore di assegnazione per CRow                                                 |
//+------------------------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- Riempie l'array con i valori
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------------------------+
//| Operatore di moltiplicazione per un'altra riga                                     |
//+------------------------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- Verifiche
   int size=Size();
   if(size!=o.Size() || size==0)
     {
      Print(__FUNCSIG__,": Fallimento nel moltiplicare le due matrici, le loro grandez
      return(res);
     }
//--- Moltiplica l'array in base agli elementi ed aggiunge i prodotti
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
```

```
//--- Risultato
   return(res);
  }
//+------------------------------------------------------------------------------+
//| Restituisce la rappresentazione in formato stringa                           |
//+------------------------------------------------------------------------------+
string CRow::String(void) const
  {
   string out="";
//--- Se la grandezza dell'array è maggiore di zero
   int size=ArraySize(m_array);
//--- Lavoriamo solo con un numero non-zero degli elementi dell'array
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- Raccoglie i valori in una stringa
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- Risultato
   return(out);
  }
//+------------------------------------------------------------------------------+
//| Classe "Matrix"                                                              |
//+------------------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- Costruttori ed un distruttore
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);             }
                    ~CMatrix(void){};
   //--- Ottiene le grandezze della matrice
   int               Rows()       const { return(ArraySize(m_rows));          }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- Restituisce i valori della colonna sottoforma di una riga CRow
   CRow              GetColumnAsRow(const int col_index) const;
   //--- Restituisce la stringa con i valori della matrice
   string            String(void) const;
   //--- L'operatore di indicizzazione restituisce la stringa per il suo numero
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));      }
   //--- Operatore addizone
   CMatrix           operator+(const CMatrix &m);
```

```
   //--- Operatore Moltiplicazione
   CMatrix           operator*(const CMatrix &m);
   //--- Operatore Assegnazione
   CMatrix          *operator=(const CMatrix &m);
  };
//+------------------------------------------------------------------------------------+
//| Un costruttore di default, crea un array di righe di grandezza zero                |
//+------------------------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- Il numero zero di righe nella matrice
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------------------------+
//| Restituisce i valori della colonna sottoforma di CRow                              |
//+------------------------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- Una variabile per ottenere i valori della colonna
   CRow row();
//--- Il numero di righe nella matrice
   int rows=Rows();
//--- Se il numero di righe è maggiore di zero, esegue l'operazione
   if(rows>0)
     {
      //--- Array per ricevere i valori della colonna con indice col_index
      double array[];
      ArrayResize(array,rows);
      //--- Filling the array
      for(int i=0;i<rows;i++)
        {
         //--- Controlla i numeri della collonna per la riga i - può eccedere i limit
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": Errore! Numero colonna ",col_index,"> grandezza riga
            break; // la riga sarà un oggetto non inizializzato
           }
         array[i]=this[i][col_index];
        }
      //--- Crea una riga CRow basata sui valori dell'array
      row=array;
     }
//--- Risultato
   return(row);
  }
//+------------------------------------------------------------------------------------+
//| Addizione di due matrici                                                           |
//+------------------------------------------------------------------------------------+
```

```
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- Il numero di righe e colonne nella matrice passata
   int cols=m.Cols();
   int rows=m.Rows();
//--- La matrice per ricevere i risultati dell'addizione
   CMatrix res(rows);
//--- Le grandezze della matrice devono corrispondere
   if(cols!=Cols() || rows!=Rows())
     {
      //--- Addizione impossibile
      Print(__FUNCSIG__,": Fallimento nell'aggiungere due matrici, le loro grandezze s
      return(res);
     }
//--- Array ausiliario
   double arr[];
   ArrayResize(arr,cols);
//--- Va attraverso le righe da aggiungere
   for(int i=0;i<rows;i++)
     {
      //--- Scrive i risultati dell'addizione delle stringe della matrice nell'array
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- Piazza l'array nella riga della matrice
      res[i]=arr;
     }
//--- restituisce il risultato di addizione delle matrici
   return(res);
  }
//+------------------------------------------------------------------------------+
//| Moltiplicazione di due matrici                                               |
//+------------------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- Numero di colonne della prima matrice, numero di righe passate nella matrice
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- Matrice per ricevere i risultati dell'addizione
   CMatrix res(rows1);
//--- Le matrici devono essere coordinate
   if(cols1!=rows2)
     {
      //--- Moltiplicazione impossibile
      Print(__FUNCSIG__,": Fallimento nel moltiplicare le due matrici, il formato è ir
            "- il numero di colonne nel primo fattore dev'essere uguale al numero di r
```

```
      return(res);
     }
//--- Array ausiliario
   double arr[];
   ArrayResize(arr,cols1);
//--- Riempie le righe nella moltiplicazione della matrice
   for(int i=0;i<rows1;i++)// Va attraverso le righe
     {
      //--- Resetta l'array di ricezione
      ArrayInitialize(arr,0);
      //--- Va attraverso gli elementi nella riga
      for(int k=0;k<cols1;k++)
        {
         //--- Prende i valori delle colonne k della matrice m nel for di CRow
         CRow column=m.GetColumnAsRow(k);
         //--- Moltiplica due righe e scrive i risultati della moltiplicazione scalare
         arr[k]=this[i]*column;
        }
      //--- piazza l'array arr[] nella i-esima riga della matrice
      res[i]=arr;
     }
//--- Restituisce il prodotto di due matrici
   return(res);
  }
//+------------------------------------------------------------------------------+
//| Operazione di Assegnazione                                                   |
//+------------------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
//--- Trova ed imposta il numero di righe
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- Riempiamo le righe con i valori di rows della matrice passata
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------------------+
//| Rappresentazione stringa della matrice                                       |
//+------------------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- Forma stringa per stringa
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
```

```
//--- Risultato
   return(out);
  }
```

Vedi anche

L' overloading, Operazioni aritmetiche, Overloading di funzioni, Regole di precedenze

Descrizione delle Funzioni Esterne

Le funzioni esterne definite in un altro modulo devono essere esplicitmente descritte. La descrizione include il tipo di ritorno, nume della funzione ed una serie di parametri di input con il loro tipi. L'assenza di tale descrizione può portare ad errori durante la compilazione, la costruzione o l'esecuzione di un programma. Quando si descrive un oggetto esterno, utilizzare la parola chiave #*Import* indicando il modulo.

Esempi:

```
#import "user32.dll"
  int     MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
  int     SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
  double  round(double value);
#import
```

Con l'aiuto di import, è facile descrivere le funzioni che sono chiamate da DLL esterne o compilato librerie EX5. Librerie EX5 sono file ex5 compilati, che hanno la proprietà library. Solo funzioni descritte con il modificatore export possono essere importate da librerie EX5.

Si prega di tenere presente che le librerie DLL ed EX5 devono avere nomi diversi (a prescindere delle directory in cui si trovano) se sono importate insieme. Tutte le funzioni importate hanno la risoluzione ambito, corrispondente al "nome file" della libreria.

Esempio:

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // chiamata del metodo della classe
      Print(::GetLastError());         // chiamata della funzione MQL5
      Print(kernel32::GetLastError()); // chiamata della funzione libreria DLL dal ke
      Print(lib::GetLastError());      // chiamata della funzione libreria EX5 da lib
     }
  };

voidOnStart()
  {
   CFoo foo;
   foo.func();
  }
```

Vedi anche

Overload, Funzioni Virtuali, Polimorfismo

Esportazione di funzioni

Una funzione dichiarata in un programma mql5 con postmodificatore*export* può essere utilizzata in un altro programma mql5. Tale funzione viene chiamata esportabile, e può essere chiamata da altri programmi dopo la compilazione.

```
int Function() export
  {
  }
```

Questo modificatore ordina al compilatore di aggiungere la funzione nella tabella di funzioni EX5 esportate da questo file EX5. Solo funzioni con tale modificatore sono accessibili ("visibile") da altri programmi mql5.

La proprietà library indica al compilatore che il file-EX5 sarà una libreria, ed il compilatore lo mostra nell'header di EX5.

Tutte le funzioni che sono previste come quelle esportabili devono essere contrassegnate con il modificatore *export*.

Vedi anche

Overload, Funzioni Virtuali, Polimorfismo

Funzioni di Gestione degli Eventi

Il linguaggio MQL5 fornisce elaborazione di alcuni eventi predefiniti. Le funzioni per la gestione di questi eventi devono essere definite in un programma MQL5; il nome della funzione, il tipo restituito, la composizione dei parametri (se ce ne sono) ed il loro tipo, devono essere rigorosamente conformeialla descrizione della funzione gestore di eventi.

L'event handler (_* gestore di eventi) del terminale client identifica le funzioni, la gestione di questo o quell'evento, dal tipo di valore di ritorno e dal tipo di parametri. Se altri parametri, non corrispondenti alle descrizioni che seguono, vengono specificati per una funzione corrispondente, o un altro tipo di ritorno è indicato per esso, una tale funzione non sarà utilizzata come un event handler.

OnStart

La funzione OnStart () è l'event handler d' Inizio, che viene generato automaticamente **solo** per l'esecuzione di **scripts**. Deve essere di tipo void, senza parametri:

```
voidOnStart()
```

Per la funzione OnStart (), può essere specificato il tipo di ritorno int.

OnInit

La funzione OnInit() è l'event handler Init. Deve essere di tipo void o int, senza parametri:

```
void OnInit();
```

L'evento Init viene generato immediatamente dopo che viene scaricato un Expert Advisor o un indicatore; questo evento non viene generato per gli script. La funzione OnInit() viene utilizzata per l'inizializzazione. Se OnInit() ha il tipo int come valore di ritorno, il codice di ritorno diverso-da-zero indica l'inizializzazione di esito negativo, e genera l'evento Deinit con il codice della ragione di deinizializzazione REASON_INITFAILED.

Per ottimizzare i parametri di ingresso di un Expert Advisor, si consiglia di utilizzare i valori dell'enumerazione ENUM_INIT_RETCODE come codice di ritorno. Questi valori sono utilizzati per organizzare il corso dell' ottimizzazione, compresa la selezione del più appropriato testing agents. Durante l'inizializzazione di un Expert Advisor prima dell'inizio del testing è possibile richiedere informazioni sulla configurazione e le risorse di un agente (il numero di cores, quantità di memoria libera, ecc.) utilizzando la funzoone TerminalInfoInteger() . Sulla base delle informazioni ottenute, ci si può permettere di usare questo agente di testing, o rifiutare di utilizzalo, per l'ottimizzazione di quell' Expert Advisor.

ENUM_INIT_RETCODE

Identificatore	Descrizione
INIT_SUCCEEDED	Inizializzazione con successo, il testing dell' Expert Advisor può essere continuato. Questo codice indica la stessa cosa del valore null - l'Expert Advisor è stato correttamente inizializzato nel tester.
INIT_FAILED	Inizializzazione fallita; non c'è alcun modo di continuare il testing a causa di errori fatali. Ad esempio, fallimento nella

Identificatore	Descrizione
	creazione di un indicatore che è richiesto per il lavoro dell' Expert Advisor. Questo valore di ritorno è lo stesso che di valore diverso da zero - l'inizializzazione dell' Expert Advisor nel tester non è riuscito.
INIT_PARAMETERS_INCORRECT	Questo valore indica l'insieme di parametri di input non corretti. La stringa risultante contenente questo codice di ritorno è evidenziata in rosso nella tabella generale di ottimizzazione. Il testing per il dato insieme di parametri dell' Expert Advisor non verrà eseguito, l'agente è libero di ricevere un nuovo compito. Alla ricezione di questo valore, lo strategy tester in modo affidabile non passerà questo compito ad altri agenti per riprovare.
INIT_AGENT_NOT_SUITABLE	Nessun errore durante l'inizializzazione, ma per qualche motivo l'agente non è adatto per il testing. Ad esempio, non c'è abbastanza memoria, non c'è Supporto OpenCL, ecc. Dopo il ritorno di questo codice, l'agente non riceverà compiti fino alla fine di questa ottimizzazione.

La funzione OnInit() del tipo void indica sempre l'inizializzazione con successo.

OnDeinit

OnDeinit() viene chiamato durante la deinizializzazione ed è l'event handler Deinit. Deve essere dichiarato come void(tipo) e dovrebbe avere un parametro di tipo const int, che contiene il codice del motivo della deinizializzazione. Se viene dichiarato un tipo diverso, il compilatore genererà un avviso, ma la funzione non verrà chiamata. Per gli script, l'evento Deinit non viene generato e quindi la funzione OnDeinit() non può essere utilizzata negli scripts.

```
void OnDeinit(const int motivo);
```

L'evento Deinit viene generato per l'Expert Advisors e gli Indicatori nei seguenti casi:

- prima della reinizializzazione dovuta alla variazione di un simbolo o di un periodo del grafico, al quale il programma MQL5 è annesso;
- prima della reinizializzazione dovuta alla variazione di parametri di input;
- prima dello scarico(_*decarico) del programma MQL5.

OnTick

L'eventoNewTick viene generato per li **Expert Advisors solo** quando viene ricevuto un nuovo tick per un simbolo, al grafico al quale è annesso l'Expert Advisor. E' inutile definire la funzione OnTick() in un indicatore personalizzato o uno script, perché l'evento NewTick non viene generato per essi.

L'evento Tick viene generato solo per Expert Advisor, ma questo non vuol dire che l'Expert Advisor ha richiesto la funzione OnTick(), dal momento che gli eventi NewTick non solo vengono generati per

Expert Advisor, ma anche eventi di Timer, BookEvent e ChartEvent vengono generati. Deve essere dichiarato come tipo void, senza parametri:

```
void OnTick();
```

OnTimer

La funzione OnTimer() viene chiamata quando avviene l'evento Timer, che viene generato dal timer di sistema solo per Expert Advisors ed Indicatori - non può essere utilizzato negli script. La frequenza del verificarsi dell'evento viene impostata al momento della sottoscrizione riguardo le notifiche su quell' evento che dev'essere ricevuto dalla funzione EventSetTimer().

È possibile annullare la sottoscrizione a ricevere gli eventi timer per un particolare Expert Advisor utilizzando la funzione EventKillTimer(). La funzione deve essere definita con il tipo void, senza parametri:

```
void OnTimer();
```

Si consiglia di chiamare la funzione EventSetTimer() una volta nella funzione OnInit(), e la funzione EventKillTimer() dovrebbe essere chiamata una volta in OnDeinit().

Ogni Expert Advisor, così come ogni Indicatore lavora con il proprio timer e riceve gli eventi solo da esso. Non appena il programma MQL5 programma si arresta, il timer viene distrutto forzatamente, se era stato stato creato ma non disabilitato dalla funzione EventKillTimer ().

OnTrade

La funzione viene chiamata quando avviene l'evento Trade, che appare quando si cambia l'elenco degli ordini piazzati e delle posizioni aperte, la cronistoria degli ordini e storia degli affari. Quando una attività di trade viene eseguita (apertura ordine pendente, apertura/chiusura posizione, impostazione stops, innesco ordini pendenti, ecc), la cronistoria degli ordini e/o elenco delle posizioni ed ordini correnti, viene cambiata di conseguenza.

```
void OnTrade();
```

Gli utenti devono autonomamente applicare nel codice, la verifica di uno stato di un account di trade, quando un tale evento viene ricevuto (se ciò è richiesto dalle condizioni della strategia di trading). Se la funzione OrderSend() chiamata è stata completata con successo e ha restituito un valore true, questo significa che il trading server ha messo l'ordine nella coda per l'esecuzione ed ha assegnato un numero di ticket ad essa. Non appena il server elabora questo ordine, l'evento Trade verrà generato. E se un utente ricorda il valore del ticket, lui/lei sarà in grado di scoprire cosa è successo all'ordine utilizzando questo valore durante l' OnTrade() event handling.

OnTradeTransaction

Quando si eseguono alcune azioni precise su un trade account, il suo stato cambia. Tali azioni comprendono:

- Inviare una richiesta di trade da qualsiasi applicazione MQL5 nel terminale client utilizzando le funzioni OrderSend e OrderSendAsync e la sua ulteriore esecuzione;
- Inviare una richiesta di trade tramite l'interfaccia grafica del terminale e la sua esecuzione ulteriore;
- Attivazione di ordini pendenti ed ordini di stop sul server;

- Esecuzione di operazioni sul lato trade server.

Le operazioni commerciali di seguito vengono eseguite come risultato di queste azioni:

- gestione di una richiesta di trade;
- cambio di ordini aperti;
- cambio della cronistoria degli ordini;
- cambio della cronistoria degli affari;
- cambio delle posizioni.

Per esempio, quando si invia un ordine di mercato buy, esso viene gestito, un appropriato ordine di buy viene creato per l'account, l'ordine viene poi eseguito e rimosso dalla lista di quelli aperti, quindi viene aggiunto alla cronistoria ordini, un appropriato affare si aggiunge alla cronistoria e una nuova posizione viene creata. Tutte queste azioni sono transazioni di trade. L' arrivo di una tale operazione presso il terminale è un evento TradeTransaction. Esso chiama l'handler(_*gestore) OnTradeTransaction

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // struttura transazione di trade
   const MqlTradeRequest&        request,      // struttura request
   const MqlTradeResult&         result        // struttura result
   );
```

L'handler contiene tre parametri:

- **trans** - questo parametro riceve la struttura MqlTradeTransaction che descrive una transazione di trade applicata ad un account;
- **request** - questo parametro riceve la struttura MqlTradeRequest che descrive una richiesta di trade;
- **result** - questo parametro riceve la struttura MqlTradeResult che descrive un risultato di esecuzione di una richiesta di trade.

Gli ultimi due parametri **request** e **result** sono riempiti da valori solo per il tipo di transazione TRADE_TRANSACTION_REQUEST, i dati sulle transazioni possono essere ricevuti dal *tipo* di parametro della variabile **trans**. Si noti che in questo caso, il campo *request_id* nella variabile **result** contiene l'ID di **request**trade request, dopo l'esecuzione di cui la transazione commerciale descritta nella variabile **trans** è stata eseguita. Request ID consente di associare l'azione eseguita (chimate di funzioni OrderSend oppure OrderSendAsync) con il risultato di questa azione inviata a OnTradeTransaction().

Una richiesta di trade manualmente inviata dal terminale o via funzioni OrderSend()/OrderSendAsync() può generare varie transazioni consecutive sul trade server. La priorità di arrivo di queste transazioni al terminale, non è garantita. Quindi, non si deve aspettare che un gruppo di transazioni arriveranno dopo un altro, durante lo sviluppo del vostro algoritmo di trading.

- Tutti i tipi di transazioni di trade sono descritti nell' enumerazione ENUM_TRADE_TRANSACTION_TYPE.
- La struttura MqlTradeTransaction descrive una transazione di trade che è riempita in modi diversi a seconda del tipo di transazione. Ad esempio, solo tipo di campo (tipo di transazione di trade) deve essere analizzato per le transazioni di tipo TRADE_TRANSACTION_REQUEST. Il secondo e terzo parametro della funzione OnTradeTransaction (richiesta e risultato) devono essere analizzati per dati aggiuntivi. Per ulteriori informazioni, vedere "Struttura di una transazione di Trade".
- Una descrizione di una transazione di trade non fornisce tutte le informazioni disponibili riguardanti gli ordini, gli affari e le posizioni (per esempio, i commenti). Le funzioni OrderGet*, HistoryOrderGet*, HistoryDealGet* and PositionGet* devono essere usate per ottenere informazioni estese.

Dopo l'applicazione di transazioni di trade per un account client, esse sono costantemente messe nella coda di transazioni del terminale, da cui costantemente vengono inviate al punto di ingresso OnTradeTransaction in ordine di arrivo al terminale.

Quando si maneggiano le transazioni di trade da parte di un Expert Advisor che utilizza l'handler OnTradeTransaction, il terminale continua a gestire le transazioni di trade che nuovamente arrivano. Pertanto, lo stato di un account di trade può già cambiare durante il funzionamento di OnTradeTransaction. Per esempio, mentre un programma MQL5 gestisce un evento di aggiunta un nuovo ordine, esso può essere eseguito, cancellato dalla lista di quelli aperti e spostato nella cronistoria. Inoltre poi, l'applicazione verrà notificata di questi eventi.

La lunghezza delle transazioni della coda comprende 1024 elementi. Se OnTradeTransaction gestisce una nuova transazione per troppo tempo, quelle vecchie in coda possono essere sostituite da quelle più recenti.

- Generalmente, non vi è rapporto preciso del numero di chiamate OnTrade e OnTradeTransaction. Una chiamata OnTrade corrisponde a una o più chiamate OnTradeTransaction.
- OnTrade viene chiamato dopo le appropriate chiamate OnTradeTransaction.

OnTester

La funzione OnTester() è il gestore dell'evento del Tester che viene generato automaticamente dopo un test di storia di un Expert Advisor dopo che un intervallo scelto è finito. La funzione deve essere definita con il tipo double, senza parametri:

```
double OnTester();
```

La funzione viene chiamata poco prima della chiamata di OnDeinit() ed ha lo stesso tipo del valore restituito - double. OnTester() può essere utilizzata solo nel testing di Expert Advisors. Il suo scopo principale è quello di calcolare un certo valore che viene utilizzato come criterio Custom max nell' ottimizzazione genetica di parametri di input.

Nell'ottimizzazione genetica viene applicato l'ordinamento discendente ai risultati entro una generazione. Cioè dal punto di vista del criterio di ottimizzazione, i migliori risultati sono quelli con valori più grandi (per i critedi di ottimizzazione Custom max vengono presi in considerazione i valori

restituiti dalla funzione OnTester). In tale ordinamento, i valori peggiori sono posizionati all'estremità e successivamente scartati e non partecipano alla formazione della prossima generazione.

OnTesterInit

La funzione OnTesterInit() è l'handler dell'evento TesterInit, che viene generato automaticamente prima dell'inizio dell' ottimizzazione di un Expert Advisor nel tester strategia. La funzione deve essere definita con il tipo void. Non ha parametri:

```
void OnTesterInit();
```

Con l'avvio dell'ottimizzazione, un Expert Advisor con l'handler OnTesterDeinit() o OnTesterPass() viene caricato automaticamente in un grafico separato del terminale con il simbolo e il periodo specificato nel tester, e riceve l'evento TesterInit. La funzione viene utilizzata per l'inizializzazione dell' Expert Advisor prima dell'inizio dell' ottimizzazione per l'ulteriore elaborazione dei risultati di ottimizzazione.

OnTesterPass

La funzione OnTesterPass() è il gestore dell'evento TesterPass, che viene generato automaticamente quando si riceve un frame durante l'ottimizzazione Expert Advisor nel tester strategia. La funzione deve essere definita con il tipo void. Non ha parametri:

```
void OnTesterPass();
```

Un Expert Advisor con l'handler OnTesterPass() viene caricato automaticamente in un grafico separato del terminale con il simbolo/periodo previsto per il testing, ed ottiene gli eventi TesterPass quando un frame viene ricevuto durante l'ottimizzazione. La funzione viene utilizzata per l'handling dinamico dei risultati di ottimizzazione "In loco" senza attendere il suo completamento. I frame vengono aggiunti utilizzando la funzione FrameAdd(), che può essere chiamata dopo la fine di una singolo pass nell' handler OnTester().

OnTesterDeinit

OnTesterDeinit() è il gestore dell'evento TesterDeinit, che viene generato automaticamente dopo la fine dell' ottimizzazione dell' Expert Advisor nel tester strategia. La funzione deve essere definita con il tipo void. Non ha parametri:

```
void OnTesterDeinit();
```

Un Expert Advisor con l'handler TesterDeinit() viene automaticamente caricato su un grafico all'inizio dell'ottimizzazione, e riceve TesterDeinit dopo il suo completamento. La funzione viene utilizzata per la lavorazione finale di tutti i risultati dell'ottimizzazione.

OnBookEvent

La funzione OnBookEvent() è l'handlerBookEvent. BookEvent viene generato per Expert Advisors ed Indicatori quando cambia il Depth of Market. Deve essere di tipo void ed avere un parametro di tipo stringa:

```
void OnBookEvent (const string& symbol);
```

Per ricevere gli eventi BookEvent per ogni simbolo, basta pre-iscriversi per ricevere questi eventi per quel simbolo con la funzione MarketBookAdd(). Per annullare l'iscrizione a ricevere gli eventi BookEvent per un simbolo particolare, chiamare MarketBookRelease().

A differenza di altri eventi, l'evento BookEvent è broadcast. Ciò significa che se un Expert Advisor sottoscrive la ricezione degli eventi BookEvent utilizzando MarketBookAdd, tutti gli altri Expert Advisors che hanno l'handler OnBookEvent() riceveranno questo evento. È pertanto necessario analizzare il nome del simbolo, che viene passato al gestore come parametro *const string& simbol*.

OnChartEvent

OnChartEvent() è l'handler di un gruppo di eventi ChartEvent:

- CHARTEVENT_KEYDOWN — evento di una sequenza di tasti, quando è focalizzata la finestra del grafico ;
- CHARTEVENT_MOUSE_MOVE — eventi di spostamento del mouse e gli eventi clic del mouse (se CHART_EVENT_MOUSE_MOVE=true è impostato per il grafico);
- CHARTEVENT_OBJECT_CREATE — evento di creazione di oggetti grafici (se CHART_EVENT_OBJECT_CREATE=true è impostato per il grafico);
- CHARTEVENT_OBJECT_CHANGE — evento di cambiamento di proprietà di un oggetto attraverso il dialogo delle proprietà;
- CHARTEVENT_OBJECT_DELETE — evento di eliminazione oggetto grafico (se CHART_EVENT_OBJECT_DELETE=true è impostato per il grafico);
- CHARTEVENT_CLICK — evento di un clic del mouse sul grafico;
- CHARTEVENT_OBJECT_CLICK — evento di un clic del mouse in un oggetto grafico appartenente al grafico;
- CHARTEVENT_OBJECT_DRAG — evento di un movimento oggetto del grafico utilizzando il mouse;
- CHARTEVENT_OBJECT_ENDEDIT — evento di modifica del testo finito nella casella di immissione dell'oggetto grafico LabelEdit;
- CHARTEVENT_CHART_CHANGE — evento di modifica del grafico;
- CHARTEVENT_CUSTOM+n — ID dell'evento dell'utente, dove n è nell'intervallo da 0 a 65535.
- CHARTEVENT_CUSTOM_LAST — l'ultimo ID accettabile di un evento personalizzato (CHARTEVENT_CUSTOM +65535).

La funzione può essere chiamata solo in Expert Advisors ed indicatori. La funzione deve essere di tipo void con 4 parametri:

```
void OnChartEvent(const int id,          // Event ID
                  const long& lparam,    // Parametro dell'evento tipo long
                  const double& dparam,  // Parametro dell'evento tipo double
                  const string& sparam   // Parametro dell'evento tipo stringa
  );
```

Per ciascun tipo di evento, i parametri di input della funzione OnChartEvent() hanno valori definiti che sono necessari per l'elaborazione di questo evento. Gli eventi e valori passati attraverso questi parametri sono elencati nella tabella seguente.

Evento	Valore del parametro id	Valore del parametro lparam	Valore del parametro dparam	Valore del parametro sparam
Evento di una sequenza di tasti	CHARTEVENT_KEYDOWN	codice di un tasto premuto	Ripete in conteggio (il numero di volte che il tasto viene ripetuto come risultato dell'utente che tiene premuto il tasto)	Il valore di stringa di una maschera di bit che descrive lo status dei pulsanti della tastiera
Eventi del mouse (Se la proprietà CHART_EVENT_MOUSE_MOVE=true è impostata per il grafico)	CHARTEVENT_MOUSE_MOVE	la coordinata X	la coordinata Y	Il valore di stringa di una maschera di bit che descrive lo stato dei pulsanti del mouse
Evento di creazione di oggetti grafici (Se CHART_EVENT_OBJECT_CREATE=true è impostato per il grafico)	CHARTEVENT_OBJECT_CREATE	—	—	Nome dell'oggetto grafico creato
Evento di cambiamento di proprietà di un oggetto attraverso la finestra delle proprietà	CHARTEVENT_OBJECT_CHANGE	—	—	Nome dell'oggetto grafico modificato
Evento di eliminazione oggetto grafico (Se CHART_EVENT_OBJECT_DELETE=true è impostato per il grafico)	CHARTEVENT_OBJECT_DELETE	—	—	Nome dell'oggetto grafico eliminato
Evento di un click del mouse sul grafico	CHARTEVENT_CLICK	la coordinata X	la coordinata Y	—
Evento di un clic del mouse in un	CHARTEVENT_OBJECT_CLICK	la coordinata X	la coordinata Y	Nome dell'oggetto

Evento	Valore del parametro id	Valore del parametro lparam	Valore del parametro dparam	Valore del parametro sparam
oggetto grafico appartenente alla tabella				grafico, in cui l'evento si è verificato
Evento di trascinamento di un oggetto grafico con il mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome dell'oggetto grafico spostato
Evento del testo finito di modifica nella casella di immissione dell'oggetto grafico LabelEdit	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome dell' oggetto grafico LabelEdit, in cui la modifica del testo è stata completata
Evento di modifiche del grafico	CHARTEVENT_CHART_CHANGE	—	—	—
ID dell'evento dell'utente con il numero N	CHARTEVENT_CUSTOM+N	Valore impostato dalla funzione EventChartCustom()	Valore impostato dalla funzione EventChartCustom()	Valore impostato dalla funzione EventChartCustom()

OnCalculate

La funzione OnCalculate() viene chiamata solo in indicatori personalizzati quando è necessario calcolare i valori degli indicatori dall'evento Calculate. Ciò si verifica tipicamente quando un nuovo tick viene ricevuto per il simbolo, per cui l'indicatore viene calcolato. Questo indicatore non è richiesto da essere collegato a qualsiasi grafico dei prezzi di questo simbolo.

La funzione OnCalculate() deve avere un tipo di ritorno int. Ci sono due possibili definizioni. All'interno di un indicatore non è possibile utilizzare entrambe le versioni della funzione.

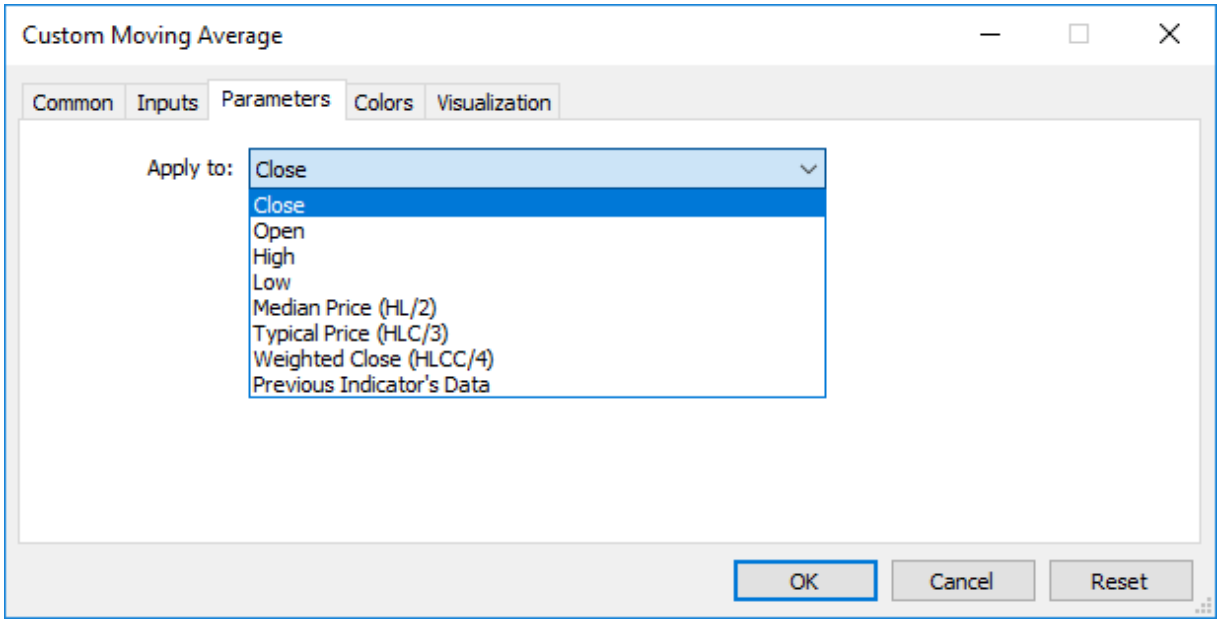
La prima forma è destinata a quegli indicatori che possono essere calcolati su un buffer di dati unico. Un esempio di tale indicatore è Custom Moving Average.

```
int OnCalculate (const int rates_total,      // size of the price[] array
                 const int prev_calculated,  // barre gestite in una chiamata preceder
                 const int begin,            // da cui partono i dati significativi
                 const double& price[]       // dall'array per il calcolo
   );
```

_*Poiché l'array prezzo[], è uno delle timeseries o un buffer calcolato di un qualche indicatore, può essere passato. Per determinare la direzione di indicizzazione dell'array price[], chiamare

ArrayGetAsSeries(). Per non dipendere dai valori di default, è necessario chiamare incondizionatamente la funzione ArraySetAsSeries() per quegli array, con cui si prevede di lavorare.

Timeseries o un indicatore necessario, per essere usato come array price[] può essere selezionato dall'utente nella scheda "Parametri" quando si avvia l'indicatore. Per fare questo, è necessario specificare l'elemento necessario nei menu a tendina del campo "Applica a".



Per ottenere valori di un indicatore personalizzato da altri programmi MQL5, viene usata la funzione iCustom(), che restituisce l'handle indicatore per le operazioni successive. È inoltre possibile specificare il prezzo l'appropriato array price[] o l'handle di un altro indicatore. Questo parametro dovrebbe essere trasmesso per ultimo nell'elenco delle variabili di input dell' indicatore personalizzato.
Esempio:

```
voidOnStart()
  {
//---
   string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
   int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
   if(handle_customMA>0)
      Print("handle_customMA = ",handle_customMA);
   else
      Print("Cannot open or not EX5 file '"+terminal_path+"\\MQL5\\Indicators\\"+"Cust
  }
```

In questo esempio, l'ultimo parametro passato è il valore PRICE_TYPICAL (dall'enumerazione ENUM_APPLIED_PRICE), che indica che l'indicatore personalizzato sarà costruito sui prezzi tipici ottenuti (High+Low+Close)/3. Se questo parametro non viene specificato, l'indicatore è costruito sulla base di valori PRICE_CLOSE, vale a dire i prezzi di chiusura di ogni barra.

Un altro esempio che mostra il passaggio dell'handler dell' indicatore come ultimo parametro per specificare l'array prezzo[], viene riportato nella descrizione della funzione iCustom ().

La seconda forma è destinata a tutti gli altri indicatori, in cui più di una time series è utilizzata per i calcoli.

```
int OnCalculate (const int rates_total,      // grandezza di input della time series
                 const int prev_calculated,  // barre gestite nella chiamata precedent
                 const datetime& time[],     // Orario
                 const double& open[],       // Open
                 const double& high[],       // High
                 const double& low[],        // Low
                 const double& close[],      // Close
                 const long& tick_volume[],  // Tick Volume
                 const long& volume[],       // Real Volume
                 const int& spread[]         // Spread
   );
```

Parametri di open[], high[], low[] e close[] contengono array con prezzi di open, prezzi high e low e prezzi close, del timeframe corrente. Il parametro time[] contiene un array con valori di tempo di apertura, il parametro spread[] ha un array contenente la cronistoria degli spreads (se qualche spread è fornito per il titolo negoziato). I parametri di volume[] e tick_volume[] contengono la cronistoria del volume di trade e tick, rispettivamente.

Per determinare la direzione di indicizzazione di time[], open[], high[], low[], close[], tick_volume[], volume[] and spread[], chiamare ArrayGetAsSeries(). Al fine di non dipendere da valori di default, si dovrebbe chiamare incondizionatamente la funzioneArraySetAsSeries() per queli array, con cui ci si aspetta di lavorare.

Il primo parametro rates_total contiene il numero di barre, disponibile per l'indicatore per il calcolo, e corrisponde al numero di barre disponibili nel grafico.

Dovremmo notare la connessione tra il valore di ritorno di OnCalculate() e il secondo parametro di input prev_calculated. Durante la chiamata di funzione, il parametro prev_calculated contiene un valore **restituito** da OnCalculate() durante la **precedente** chiamata. Questo permette agli algoritmi economici di fare calcoli con l'indicatore personalizzato al fine di evitare calcoli ripetuti per quelle barre che non sono state modificati sin dalla precedente esecuzione di questa funzione.

Per questo, è di solito sufficiente restituire il valore del parametro rates_total, che contiene il numero di barre nella chiamata alla funzione in corso. Se dopo l'ultima chiamata di OnCalculate() i dati sui prezzi sono cambiati (uno storico più profonda scaricato o spazi di storico riempiti di vuoti), il valore del parametro di input prev_calculated sarà impostato a zero dal terminale.

Nota: se OnCalculate restituisce zero, allora i valori dell'indicatore non sono mostrati nella DataWindow del terminale client.

Per capire meglio, sarebbe utile aviare l'indicatore, il cui codice è allegato qui sotto.

Esempio Indicatore:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot linea
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
```

```
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers indicatore
double         LineBuffer[];
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
  {
//--- Prende il numero di barre disponibili per il simbolo corrente e periodo del gra
   int bars=Bars(Symbol(),0);
   Print("Bars = ",bars,", rates_total = ",rates_total,",  prev_calculated = ",prev_ca
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Eseguire Programmi, Eventi Terminale Client, Utilizzo degli eventi

Variabili

Dichiarazione delle variabili

Le variabili devono essere dichiarate prima di essere usate. Nomi unici vengono usati per identificare variabili. Per dichiarare una variabile, è necessario specificare il tipo ed il nome univoco. Dichiarazione di variabile non è un operatore.

Tipi semplici sono:

- char, short, int, long, uchar, ushort, uint, ulong - integers;
- color - numero intero che rappresenta il colore-RGB;
- datetime - la data e l'ora, un numero intero senza segno che contiene il numero di secondi dall'ora 0 del 1 gennaio 1970;
- bool - valori booleani *true* e *false*;
- double - numero in virgola mobile a doppia-precisione;
- float - numero in virgola mobile a precisione-singola;
- string - stringhe di caratteri.

Esempi:

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

I tipi complessi o composti:

Le strutture sono tipi di dati composti, costruiti utilizzando altri tipi.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // Variabile della precedentemente dichiarata struttura MyTime
```

Non è possibile dichiarare variabili del tipo di struttura fino a che non si dichiarara la struttura.

Arrays

L' Array è la sequenza indicizzata di tipi di dato identici:

```
int    a[50];       // Array uni-dimensionale di 50 interi.
double m[7][50];    // Array bi-dimensionale di sette array,
                    // ciascuno composto da 50 numeri.
MyTime t[100];      // array contenente elementi come MyTime
```

Solo un numero intero può essere un indice di array. Non più di quattro dimensioni array sono ammesse. La numerazione degli elementi inizia con 0. L'ultimo elemento di una matrice uni-dimensionale ha il numero che è di 1 inferiore alla grandezza della matrice. Ciò significa che la chiamata per l'ultimo elemento di un array composto da 50 numeri interi apparirà come a[49]. La stessa cosa vale per array multidimensionali: Una dimensione è indicizzata da 0 fino alla grandezza della dimensione -1. L'ultimo elemento di un array bi-dimensionale dall'esempio apparirà come m[6][49].

Array statici non possono essere rappresentati come timeseries, ad es. la funzione ArraySetAsSeries(), che definisce l'accesso ad elementi dell'array dalla fine all'inizio, non può essere applicata ad essi. Se si desidera fornire l'accesso ad un array così come nelle timeseries, utilizzare l'oggetto array dinamico.

Se vi è un tentativo di accesso di fuori del range dell'array, il sottosistema di esecuzione genererà un errore critico ed il programma verrà interrotto.

Metodi integrati per lavorare con gli array

Le funzioni della sezione Funzioni con gli Array, così come i metodi integrati possono essere utilizzati per gestire gli array:

Metodo	Analogo	Descrizione
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill, ArrayInitialize	Riempie l'array con il valore specificato
void array.Free();	ArrayFree	Rilascia il buffer dell'array dinamico e imposta la dimensione zero a 0 (zero)
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	Imposta una nuova grandezza nella prima dimensione dell'array
int array.Print();	ArrayPrint	Visualizza i valori di un array di tipo semplice nel journal
int array.Size(const int range=-1);	ArraySize, ArrayRange	Restituisce il numero di elementi dell'intero array (range=-1) o della dimensione dell'array specificata
bool array.IsDynamic();	ArrayIsDynamic	Verifica se l'array è dinamico
bool array.IsIndicatorBuffer();		Verifica se l'array è un buffer di un indicatore
bool array.IsSeries();	ArrayIsSeries	Verifica se l'array è una timeseries
bool array.AsSeries();	ArrayGetAsSeries	Controlla la direzione di indicizzazione dell'array
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	Imposta la direzione di indicizzazione nell'array

Metodo	Analogo	Descrizione
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	Copia i valori dell'array in un altro array
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	Restituisce il risultato del confronto di due array di tipo semplice o di strutture personalizzate
int array.Insert(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayInsert	Inserisce il numero specificato di elementi da un array sorgente a uno ricevente partendo da un indice specificato
int array.Remove(const int start_pos, const int count);	ArrayRemove	Rimuove il numero specificato di elementi dall'array partendo dall'indice specificato
int array.Reverse(const int start_pos, const int count);	ArrayReverse	Inverte il numero specificato di elementi nell'array partendo dall'indice specificato
bool array.Swap(array& arr[]);	ArraySwap	Scambia il contenuto con un altro array dinamico dello stesso tipo
void array.Sort(sort_function);	ArraySort	Ordina gli array numerici in base alla prima dimensione
int array.Search(scalar value, search_function);	ArrayBsearch	Restituisce l'indice del primo elemento individuato nella prima dimensione dell'array
int array.Find((scalar value, search_function);		Esegue una ricerca nell'array utilizzando la funzione passata e restituisce l'indice del primo elemento rilevato
array array.Select(scalar value, search_function);		Esegue una ricerca nell'array utilizzando la funzione passata e restituisce l'array con tutti gli elementi rilevati

Specificatori di accesso

Specificatori di accesso definiscono come il compilatore può accedere a variabili, membri di strutture o classi.

Lo specificatore const dichiara una variabile come una costante, e non consente di modificare questa variabile durante l'esecuzione. E' consentita una singola inizializzazione di una variabile quando questa si dichiara.

Esempio:

```
int OnCalculate (const int rates_total,      // size of the price[] array
                 const int prev_calculated,  // barre gestite in una chiamata precedent
                 const int begin,            // da cui partono i dati significativi
                 const double& price[]       // dall'array per il calcolo
   );
```

Per accedere ai membri di strutture e classi utilizzare i seguenti qualificatori:

- public - consente l'accesso illimitato alla variabile o al metodo della classe
- protected - permette l'accesso da metodi di questa classe, come pure da metodi di classipubblicamente ereditate. Altro accesso è impossibile;
- private - consente l'accesso a variabili e metodi della classe solo da metodi della classe stessa.
- virtual - si applica solo ai metodi della classe (ma non ai metodi di strutture) e dice al compilatore che il metodo deve essere inserito nella tabella delle funzioni virtuali della classe.

Classi di memorizzazione

Ci sono tre classi di memorizzazione: static, input ed extern. Questi modificatori di una classe di memorizzazione indicano esplicitamente al compilatore che le variabili corrispondenti sono distribuite in un area di memoria pre-assegnata, che è chiamata il pool globale. Inoltre, questi modificatori indicano il trattamento speciale dei dati delle variabili. Se una variabile dichiarata a livello locale non è una static , la memoria per tale variabile viene assegnata automaticamente allo stack programma. La liberazione della memoria allocata per un array non-statico viene anche eseguita automaticamente quando si va oltre l'area di visibilità del blocco, in cui è dichiarato l'array.

Vedi anche

Tipi di dati, Incapsulamento ed estensibilità dei Tipi,Inizializzazione delle Variabili, Visibilità Campo di Applicazione e Durata delle Variabili, Creazione ed Eliminazione di Oggetti, Membri Statici di una Classe

Variabili locali

Una variabile dichiarata all'interno di una funzione è locale. L'ambito di una variabile locale è limitato alla gamma di funzioni all'interno del quale è stata dichiarata. La variabile locale può essere inizializzata dal risultato di ogni espressione. Ogni chiamata della funzione inizializza una variabile locale. Le variabili locali vengono memorizzate in un'area di memoria, della funzione corrispondente.

Esempio:

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

La visibilità di una variabile è la parte del programma, in cui una variabile può essere riferita. Le variabili dichiarate all'interno di un blocco (sul piano interno), hanno il blocco come loro campo di applicazione. Il campo di applicazione blocco inizia con la dichiarazione della variabile e termina con la parentesi graffa finale destra.

Le variabili locali dichiarate all'inizio di una funzione hanno anche la visibilità del blocco, così come i parametri di funzioni che sono variabili locali. Ogni blocco può contenere dichiarazioni di variabili. Se i blocchi sono nidificati e l' identificatore nel blocco esterno ha lo stesso nome dell'identificatore nel blocco interno, l'identificatore del blocco esterno è nascosto, finché il funzionamento del blocco interno non è finito.

Esempio:

```
voidOnStart()
  {
//---
   int i=5;       // variabile locale della funzione
     {
      int i=10;  // variabile della funzione
      Print("Inside block i = ",i); // il risultato è i=10;
     }
   Print("Blocco esterno i = ",i);  // il risultato è  i=5;
  }
```

Ciò significa che mentre il blocco interno è in esecuzione, esso vede i valori dei propri identificatori locali e non i valori di identificatori con nomi identici nel blocco esterno.

Esempio:

```
voidOnStart()
  {
//---
   int i=5;       // variabile locale della funzione
   for(int i=0;i<3;i++)
      Print("Interno per i = ",i);
   Print("Esterno per il blocco i = ",i);
```

```
  }
/* Risultato dell'esecuzione
Interno per i = 0
Interno per i = 1
Interno per i = 2
Blocco esterno i = 5
*/
```

Le variabili locali dichiarate come static hanno la visibilità del blocco, nonostante il fatto che esse esistono sin dal momento in cui il programma inizia.

Stack

In ogni programma MQL5, un'area di memoria speciale chiamata stack viene allocata per memorizzare le variabili locali delle funzioni che vengono create automaticamente. Uno stack viene allocato per tutte le funzioni, le dimensioni predefinite per indicatori sono pari a 1 Mb. In Expert Advisors e gli script, dimensione dello stack può essere gestito tramite il #property stacksize istruzione di compilatore (che imposta la dimensione dello stack in byte), una memoria di 8Mb viene allocata di default per la pila.

Le variabili locali Static sono memorizzate nello stesso luogo dove altre variabili static e global vengono memorizzate - in un'area di memoria speciale, che esiste separatamente dallo stack. Le variabili createDinamicamente usano anche un'area di memoria separata dallo stack.

Ad ogni chiamata di funzione, un posto sullo stack viene allocato per variabili interne non-statiche. Dopo l'uscita dalla funzione, la memoria è disponibile per l'uso.

Se dalla prima funzione viene chiamata la seconda, allora la seconda funzione occupa la grandezza richiesta dalla memoria stack rimanente per le sue variabili. Pertanto, quando si utilizzano funzioni incluse, lo stack di memoria sarà occupato sequenzialmente per ogni funzione. Questo può portare ad una carenza di memoria durante una delle chiamate di funzione, tale situazione è chiamata stack overflow.

Pertanto, per i dati locali grandi, si dovrebbe meglio usare la memoria dinamica - quando si entra in una funzione, allocare la memoria, che è richiesta per le esigenze locali, nel sistema (new, ArrayResize()), ed all'uscita della funzione, rilasciare la memoria (delete, ArrayFree()).

Vedi anche

Tipi di dati, Incapsulamento ed Estensibilità dei Tipi,Inizializzazione delle Variabili, Visibilità Campo di Applicazione e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Parametri Formali

I parametri passati alla funzione sono locali. Il campo di applicazione è il blocco della funzione. Parametri formali devono avere nomi diversi da quelli delle variabili esterne e variabili locali definite all'interno di una funzione. Alcuni valori possono essere assegnati ai parametri formali nel blocco della funzione. Se un parametro formale viene dichiarato con il modificatore const, il suo valore non può essere modificato all'interno della funzione.

Esempio:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

Parametri formali possono essere inizializzati da costanti. In questo caso, il valore di inizializzazione viene considerato come valore predefinito. Parametri, accanto a quelli inizializzati, devono anch'essere inizializzati.

Esempio:

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

Quando si chiama una funzione, i parametri inizializzati possono essere omessi, quelli di default vengono sostituiti al loro posto.

Esempio:

```
func(123, 0.5);
```

Parametri di tipo semplice sono passati per valore, ad. es., modifiche della corrispondente variabile locale di questo tipo all'interno della funzione chiamata non si rifletteranno nella funzione chiamante. Array di qualsiasi tipo e dati del tipo struttura, vengono sempre passati per riferimento. Se è necessario vietare, modificando il contenuto dell'array o struttura, i parametri di questo tipo devono essere dichiarati con la parola chiave *const*.

Vi è la possibilità di passare i parametri di tipo semplice, per riferimento. In questo caso, la modifica di tali parametri all'interno della funzione chiamante influenzerà le corrispondenti variabili passate per riferimento. Per indicare che un parametro viene passato per riferimento, mettere il modificatore & dopo il tipo di dato.

Esempio:

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
   for(int i=0; i<OrdersTotal(); i++)
```

```
     {
      if(i==ArraySize(z))        break;
      if(OrderSelect(i)==false) break;
      z[i]=OrderOpenPrice();
     }
   x=i;
   y=calculated_tp;
  }
```

I parametri passati per riferimento, non possono essere inizializzati con i valori predefiniti.

Un massimo di 64 parametri può essere passato in una funzione.

Vedi anche

Variabili di Input, Tipi di dato, Incapsulamento ed Estensibilità dei Tipi, Inizializzazione delle Variabili, Visibilità Campo di Applicazione e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Variabili Statiche

La classe di memoria di static definisce una variabile statica. Il modificatore static viene indicato prima del tipo di dati.

Esempio:

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

Una variabile static può essere inizializzata da una costante o espressione costante corrispondente al suo tipo, a differenza di una semplice variabile locale, che può essere inizializzata da qualsiasi espressione.

Variabili statiche esistono dal momento di esecuzione del programma e vengono inizializzate solo una volta prima che le funzioni specializzate OnInit() vengano chiamate. Se i valori iniziali non vengono specificati, le variabili della classe di memorizzazione static stanno assumendo i valori zero iniziali.

Le variabili locali dichiarate con la parola chiave *static* mantengono i loro valori per tutto il lifetime della funzione . Ad ogni chiamata di funzione successiva, tali variabili locali contengono i valori che avevano durante la chiamata precedente.

Tutte le variabili in un blocco, ad eccezione di parametri formali di una funzione, possono essere definiti come statico. Se una variabile dichiarata a livello locale non è una statica, la memoria per tale variabile viene assegnata automaticamente ad uno stack di programma.

Esempio:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("La fuzione Counter viene chiamata",count," times");
   return count;
  }
voidOnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c =",c);
  }
```

Vedi anche

Tipi di Dati, Incapsulamento ed Estensibilità dei Tipi,Inizializzazione delle Variabili, Ambito Visibilità e Durata delle Variabili, Creazione ed Eliminazione di Oggetti,Membri di Classi Statiche

Variabili Globali

Le variabili globali vengono create mettendo le loro dichiarazioni fuori dalle descrizioni delle funzioni. Variabili globali sono definite allo stesso livello come funzioni, cioè, non sono locali in ogni blocco.

Esempio:

```
int GlobalFlag=10;   // Variabile globale
int OnStart()
  {
   ...
  }
```

Il campo di applicazione delle variabili globali è l'intero programma. Le variabili globali sono accessibili da tutte le funzioni definite nel programma. Esse vengono inizializzate a zero a meno che un altro valore iniziale non viene definito in modo esplicito. Una variabile globale può essere inizializzata solo da una costante o un'espressione che corrisponde al suo tipo.

Global variables are initialized only once after the program is loaded into the client terminal memory and before the first handling of the Init event. For global variables representing class objects, during their initialization the corresponding constructors are called. In scripts global variables are initialized before handling the Start event.

Nota: Le variabili dichiarate a livello globale non devono essere mischiate con le variabili globali del terminale client a cui è possibile accedere utilizzando le funzioni GlobalVariable...().

Vedi anche

Tipi di dati, Incapsulamento ed Estensibilità dei Tipi,Inizializzazione delle Variabili, Visibilità Campo di Applicazione e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Variabili di Input

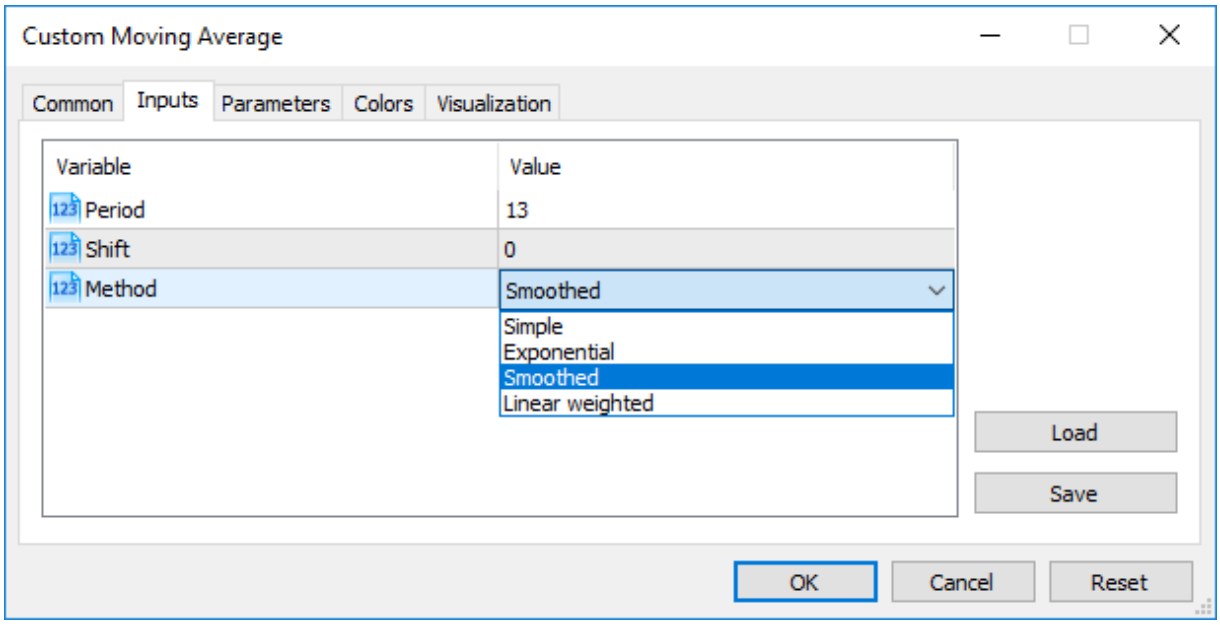
La classe di memorizzazione input definisce la variabile esterna. Il modificatore *input* viene indicato prima del tipo di dati. Una variabile con il modificatore input non può essere modificata all'interno dei programmi MQL5, tali variabili sono accessibili in sola lettura. I valori delle variabili di ingresso possono essere modificati solo dall'utente dalla finestra delle proprietà del programma. Le variabili esterne vengono sempre reinizializzate immediatamente prima che OnInit() venga chiamato.

The maximum length of input variable names is 63 characters. For the input parameter of string type, the **maximum** value length (string length) can be from 191 to 253 characters (see the Note). The minimum length is 0 characters (the value is not set).

Esempio:

```
//--- parametri di input
input int             MA_Period=13;
input int             MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

Variabili di input determinano i parametri di input di un programma. Esse sono disponibili nella finestra Proprietà di un programma.

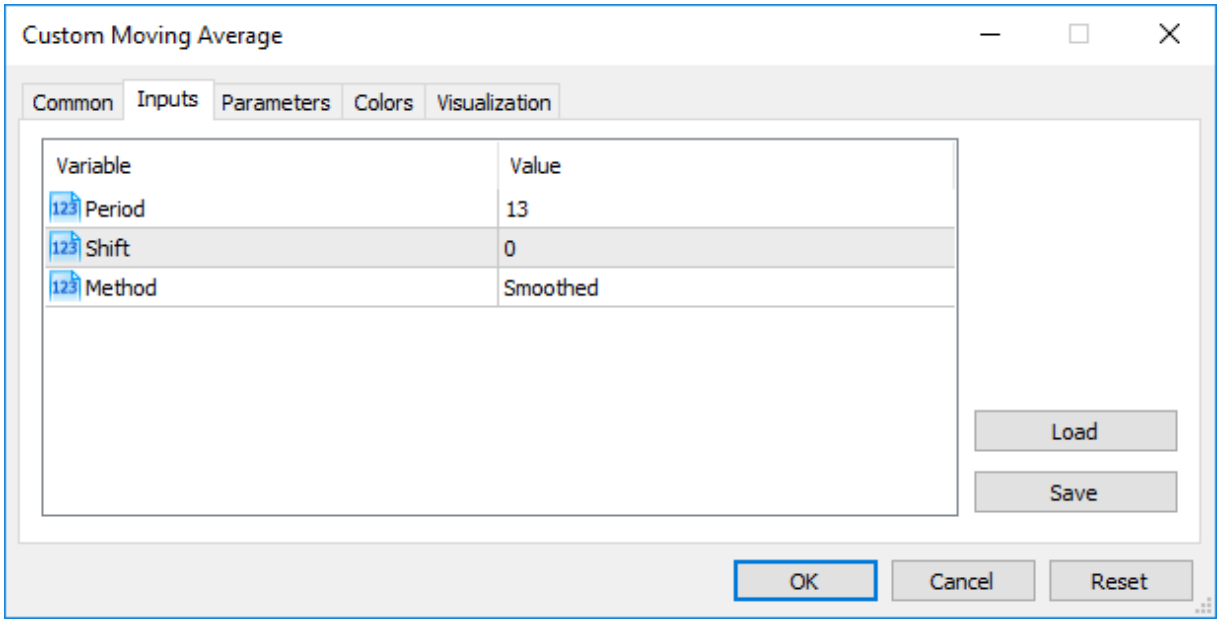


È possibile impostare un altro modo per visualizzare i nomi dei parametri di input nella scheda Input. Per fare questo, viene utilizzato un commento stringa, che dovrebbe trovarsi dopo la descrizione di un parametro di input nella stessa riga. Così, nomi più comprensibili per un utente possono essere abbinati a parametri di input.

Esempio:

```
//--- parametri di input
input int             InpMAPeriod=13;         // Periodo di smussamento
input int             InpMAShift=0;           // Slittamento linea orizziontale
```

```
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // Metodo di smussamento
```



Nota: Array e variabili di tipo complesso non possono agire come variabili di input.

Nota: La lunghezza di un di un commento string per le variabili di input non può superare i 63 caratteri.

Note: For input variables of string type, the limitation of the value length (string length) is set by the following conditions:

- the parameter value is represented by the "parameter_name=parameter_value" string ('=' is considered),
- maximum representation length of 255 characters (*total_length_max*=255 or 254 characters excluding '='),
- maximum length of the *parameter_name_length* string parameter = 63 characters.

Thus, the maximum string size for a string parameter is calculated using the equation:

```
parameter_value_length=total_length_max-parameter_name_length=254-parameter_name_lengt
```

This provides the maximum string size from 191 (parameter_name_length=63) to 253 characters (parameter_name_length=1).

Passaggio di parametri al momento della chiamata Indicatori personalizzati da Programmi MQL5

Gli indicatori personalizzati vengono chiamati con la funzione iCustom(). Dopo il nome dell' indicatore personalizzato, i parametri dovrebbero andare in stretta conformità con la dichiarazione delle variabili di input di questo indicatore personalizzato. Se i parametri indicati sono meno delle variabili di input dichiarate nell'indicatore personalizzato chiamato, i parametri mancanti vengono riempiti con i valori specificati durante la dichiarazione delle variabili.

Se l'indicatore personalizzato utilizza la funzione OnCalculate del primo tipo (cioè, l'indicatore è calcolato utilizzando lo stesso array di dati), allora uno dei valori ENUM_APPLIED_PRICE o l'handle di un altro indicatore deve essere utilizzato come ultimo parametro quando si chiama un indicatore personalizzato. Tutti i parametri relativi alle variabili di input devono essere chiaramente indicati.

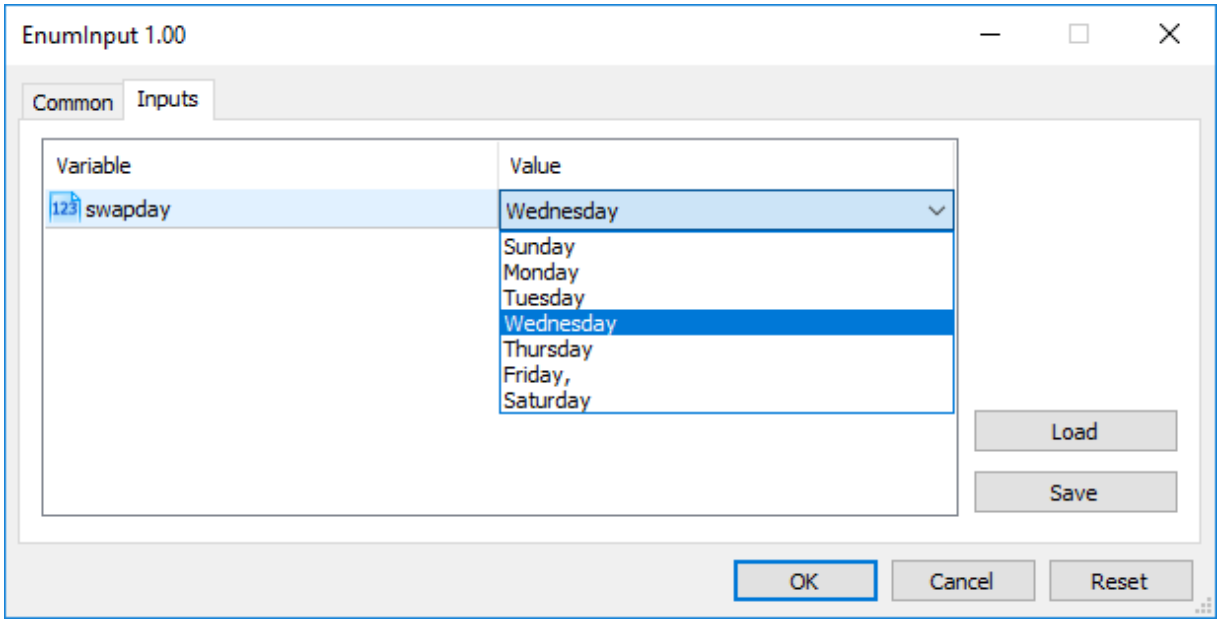
Enumerazioni come Parametri di Input

Non solo le enumerazioni built-in fornite in MQL5, ma anche variabili definite dall'utente possono essere utilizzate come variabili di input (parametri di input per programmi MQL5). Per esempio, si può creare l'enumerazione dayOfWeek, descrivendo giorni della settimana, ed utilizzare la variabile di input per specificare un particolare giorno della settimana, non come un numero, ma in modo più comune.

Esempio:

```
#property script_show_inputs
//--- giorno della settimana
enum dayOfWeek
  {
   S=0,      // Domenica
   M=1,      // Lunedì
   T=2,      // Martedì
   W=3,      // Mercoledì
   Th=4,     // giovedì
   Fr=5,     // Venerdì,
   St=6,     // Sabato
  };
//--- parametri di input
input dayOfWeek swapday=W;
```

Al fine di consentire ad un utente di selezionare un valore necessario dalla finestra delle proprietà durante l'avvio dello script, si usa il comando del preprocessore #property script_show_inputs. Avviamo lo script e possiamo scegliere uno dei valori dell'enumerazione dayOfWeek dall'elenco. Avviamo lo script EnumInInput ed andiamo alla alla scheda Input. Per impostazione predefinita, il valore di swapday (giorno di triplo carico swap) è Mercoledì (W=3), ma è possibile specificare un altro valore, ed utilizzare questo valore per modificare il funzionamento del programma.



Numero di possibili valori di un'enumerazione è limitato. Per selezionare un valore di input viene utilizzata una lista a discesa. Nomi mnemonici dei membri di enumerazione vengono utilizzati per i valori visualizzati nella lista. Se un commento è associato ad un nome mnemonico, come mostrato in questo esempio, il contenuto del commento viene usato al posto del nome mnemonico.

Ogni valore dell'enumerazione dayOfWeek ha il suo valore da 0 a 6, ma nella lista dei parametri, verranno mostrati i commenti specificati per ogni valore. Ciò fornisce una maggiore flessibilità per la scrittura di programmi con chiare descrizioni di parametri di input.

Le variabili con Modificatore sinput

Le variabili con il modificatore input consentono non solo l'impostazione di valori dei parametri esterni al momento del lancio dei programmi, ma sono anche necessarie per ottimizzare le strategie di trading nello Strategy Tester. Ogni variabile di input esclusa quella del tipo stringa, può essere utilizzata nell' ottimizzazione.

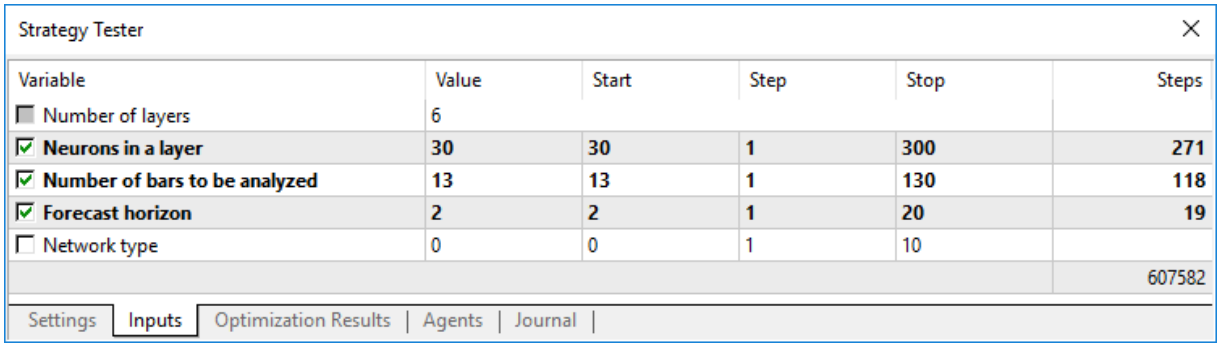
Talvolta, è necessario escludere alcuni parametri esterni di programma dall'area di tutti i passi nel tester. sinput è un modificatore di memoria che è stato introdotto per casi del genere. sinput sta per dichiarazione statica di variabile esterna (sinput = static input). Ciò significa che la seguente dichiarazione in un codice di Expert Advisor

```
sinput       int layers=6;    // Numero di strati
```

sarà equivalente alla dichiarazione completa

```
static input int layers=6;    // Numero di strati
```

La variabile dichiarata con modificatore sinput è un parametro di input di un programma MQL5. Il valore di questo parametro può essere modificato al momento del lancio del programma. Tuttavia, questa variabile non viene utilizzata per l'ottimizzazione dei parametri di input. In altre parole, i suoi valori non vengono enumerati quando si va alla ricerca del miglior set di parametri che corrispondono ad una condizione specificata.



L'Expert Advisor mostrato sopra ha 5 parametri esterni. "Numero di strati" è dichiarato in modo da essere sinput e pari a 6. Questo parametro non può essere modificato durante l'ottimizzazione della strategia di trading. Possiamo specificare il valore necessario per esso, per essere utilizzato ulteriormente. I campi Start, Step e Stop non sono disponibili per tale variabile.

Pertanto, gli utenti non saranno in grado di ottimizzare questo parametro dopo aver specificato il modificatore sinput per la variabile. In altre parole, gli utenti dei terminali non saranno in grado di impostare i valori iniziali e finali nello Strategy Tester per l'enumerazione automatica nell'intervallo specificato, durante l'ottimizzazione.

Tuttavia, vi è una sola eccezione a questa regola: le variabili sinput possono essere variate in attività di ottimizzazione utilizzando la funzione ParameterSetRange(). Questa funzione è stata introdotta specificatamente per il controllo del programma sui sets di valori disponibili per qualsiasi variabile input comprese quelle dichiarate come static input (sinput). La funzione ParameterGetRange() consente di ricevere i valori delle variabili di input quando viene lanciata l'ottimizzazione (nell' handler OnTesterInit()) e per resettare un valore di cambiamento di step, all'interno del quale, i valori dei parametri ottimizzati verranno enumerati.

In questo modo, combinando il modificatore sinput e due funzioni che lavorano con parametri di input, si consente di creare un setting di regole flessibili per l'impostazione di intervalli di ottimizzazione dei parametri di input che dipendono da valori di altri parametri di input.

Organizzazione dei parametri di input

Per la comodità di lavoro con i programmi MQL5, i parametri di input possono essere divisi in blocchi nominativi, usando la parola chiave group. Ciò consente la separazione visiva di alcuni parametri da altri in base alla logica in essi incorporata.

```
input group             "Group name"
input int               variable1 = ...
input double            variable2 = ...
input double            variable3= ...
```

Dopo tale dichiarazione, tutti i parametri di input vengono uniti visivamente in un gruppo specificato, semplificando la configurazione dei parametri per gli utenti MQL5 all'avvio su un chart o nello strategy tester. Le specifiche di ciascun gruppo sono valide fino a quando appare una nuova dichiarazione di gruppo:

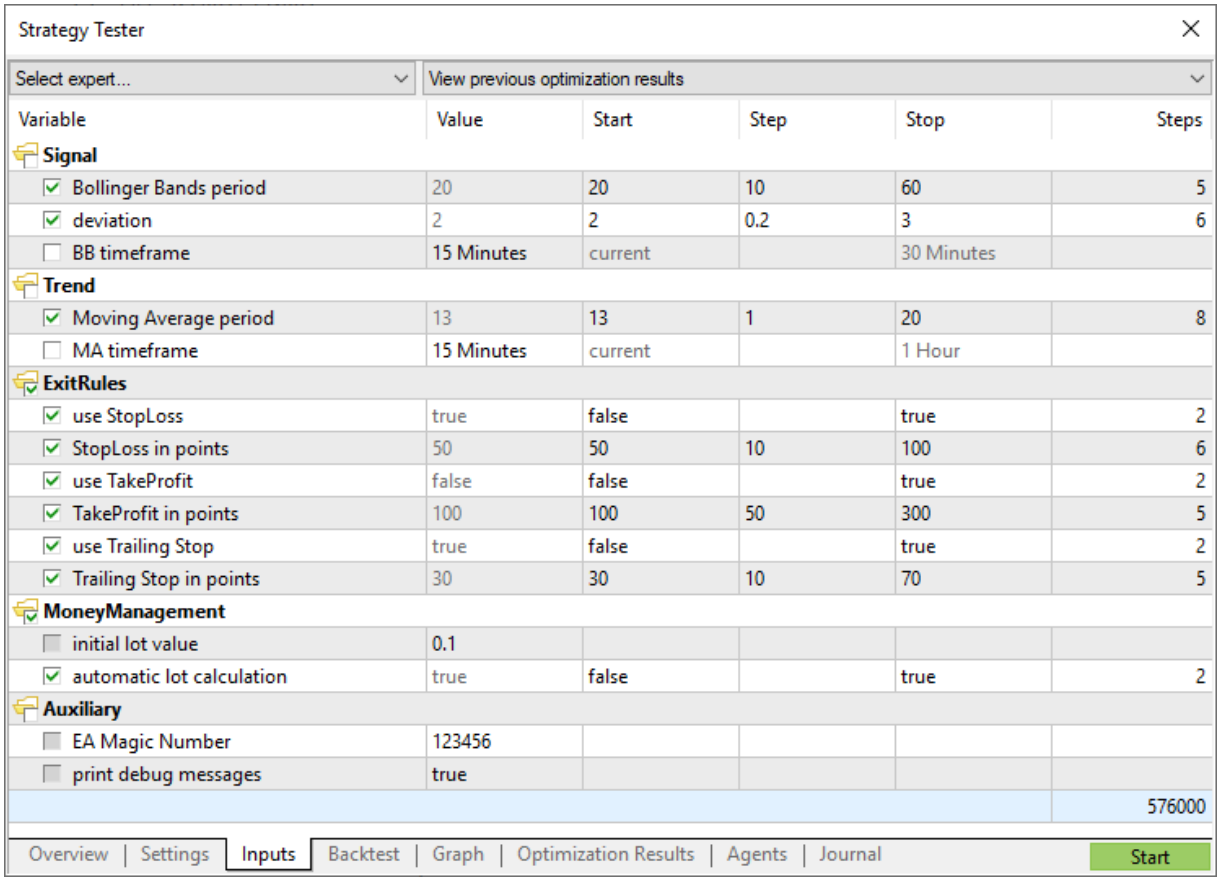
```
input group             "Nome gruppo #1"
input int               group1_var1 = ...
```

```
input double          group1_var2 = ...
input double          group1_var3 = ...

input group           "Nome gruppo #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

Un EA di esempio con i blocchi di input separati dal loro scopo:

```
input group           "Signal"
input int             ExtBBPeriod   = 20;       // Bollinger Bands period
input double          ExtBBDeviation= 2.0;      // deviazione
input ENUM_TIMEFRAMES ExtSignalTF=PERIOD_M15;   // BB timeframe

input group           "Trend"
input int             ExtMAPeriod   = 13;       // Moving Average period
input ENUM_TIMEFRAMES ExtTrendTF=PERIOD_M15;    // MA timeframe

input group           "ExitRules"
input bool            ExtUseSL      = true;     // usa StopLoss
input int             Ext_SL_Points = 50;       // StopLoss in punti
input bool            ExtUseTP      = false;    // usa TakeProfit
input int             Ext_TP_Points = 100;      // TakeProfit in punti
input bool            ExtUseTS      = true;     // usa Trailing Stop
input int             Ext_TS_Points = 30;       // Trailing Stop in punti

input group           "MoneyManagement"
sinput double         ExtInitialLot = 0.1;      // valore lotto iniziale
input bool            ExtUseAutoLot = true;     // calcolo automatico del lotto

input group           "Auxiliary"
sinput int            ExtMagicNumber = 123456;  // EA Magic Number
sinput bool           ExtDebugMessage= true;    // stampa messaggi di debug
```

Quando si avvia tale EA nello strategy tester, è possibile fare doppio clic sul nome di un gruppo per comprimere/espandere il blocco di input, nonché fare clic sulla casella di controllo del gruppo per selezionare tutti i suoi parametri per l'ottimizzazione.



Vedi anche

iCustom, Enumerazioni, Proprietà dei Programmi

Variabili Esterne

La parola chiave extern viene utilizzata per dichiarare identificatori di variabili come identificatori della classe staticadi memorizzazione con lifetime globale. Queste variabili esistono dall'inizio del programma e la memoria per esse viene allocata ed inizializzata immediatamente dopo l'inizio del programma.

È possibile creare programmi composti da più file di sorgenti, in questo caso viene utilizzata la direttiva per il preprocessore #include. Le variabili dichiarate come extern con lo stesso tipo e l'identificatore possono esistere in diversi file sorgente di un progetto.

Quando si compila il progetto, tutte le variabili extern con lo stesso tipo ed un identificatore sono associate con una parte di memoria del pool di variabile globale. Variabili extern sono utili per la compilazione separata di file di origine. Variabili esterne possono essere inizializzate, ma soltanto una volta - l'esistenza di diverse variabili extern inizializzate dello stesso tipo e con lo stesso identificatore, è vietata.

Vedi anche

Tipi di dati, Incapsulamento ed Estensibilità dei Tipi,Inizializzazione delle Variabili, Visibilità Campo di Applicazione e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Inizializzazione delle Variabili

Ogni variabile può essere inizializzata in fase di definizione. Se una variabile non è inizializzata esplicitamente, il valore memorizzato in questa variabile può essere qualunque. L' inizializzazione implicita non viene utilizzata.

Variabili global e static possono essere inizializzate solo da una costante del tipo corrispondente o un'espressione costante. Le variabili locali possono essere inizializzate da qualsiasi espressione, non solo una costante.

L' inizializzazione delle variabili globali e statiche viene eseguita una sola volta. L' inizializzazione delle variabili locali viene fatta ogni volta che si chiama la funzione corrispondente.

Esempi:

```
int    n       = 1;
string s       = "hello";
double f[]     = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4 } };
//--- dal tetris
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- inizializzazione di tutti i campi delle strutture con valori zero
MqlTradeRequest request={};
```

La lista dei valori degli elementi di un array deve essere racchiusa tra parentesi graffe. Le sequenze di inizializzazione perse sono considerate uguali a 0.

Se la dimensione della matrice inizializzata non è specificata, viene determinata da un compilatore, in base alla dimensione della sequenza di inizializzazione.

Esempi:

```
struct str3
  {
   int               low_part;
   int               high_part;
  };
struct str10
  {
   str3              s3;
   double            d1[10];
   int               i3;
  };
voidOnStart()
  {
   str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
   str10 s10_2={{1,0},{},100};
   str10 s10_3={{1,0},{1.0}};
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
```

```
   Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
   Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
   Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
  }
```

Per la variabile di tipo struttura, l'inizializzazione parziale è consentita, così come per gli array statici (con una grandezza implicitamente impostata). È possibile inizializzare uno o più elementi prima di una struttura o un array, gli altri elementi vengono inizializzati con la cifra zero in questo caso.

Vedi anche

Tipi di Dati, Incapsulamento ed Estensibilità dei Tipi, Visibilità Campo di applicazione e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Visibilità Ambito(_*scope) e Durata delle Variabili(_*lifetime)

Ci sono due tipi fondamentali di ambito: ambito locale ed ambito globale.

Una variabile dichiarata al di fuori di tutte le funzioni si trova in ambito globale. L'accesso a tali variabili può essere fatto da ovunque nel programma. Queste si trovano nel pool globale di memoria, per cui la loro vita coincide con la durata del programma.

Una variabile dichiarata all'interno di un blocco (parte del codice racchiuso tra parentesi graffe) appartiene all' ambito locale. Tale variabile non è visibile (e quindi non disponibile) fuori del blocco, in cui è dichiarata. Il caso più comune di dichiarazione locale è una variabile dichiarata all'interno di una funzione. Una variabile dichiarata localmente, si trova sullo stack, e la durata di tale variabile è uguale alla durata della funzione.

Poiché l'ambito di una variabile locale è il blocco in cui è dichiarata, è possibile dichiarare variabili con lo stesso nome, come quelle di variabili dichiarate in altri blocchi, come pure quelle dichiarate a livelli superiori, fino a livello globale.

Esempio:

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[]
  {
   int        i,limit;
   static int weightsum=0;
   double     sum=0;
//---
   if(prev_calculated==0)
     {
      limit=MA_Period+begin;
      //--- imposta il valore vuoto per la prima barra limite
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- calcola il primo valore visibile
      double firstValue=0;
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
      firstValue/=(double)weightsum;
      LineBuffer[limit-1]=firstValue;
     }
   else
     {
      limit=prev_calculated-1;
     }

   for(i=limit;i<rates_total;i++)
     {
      sum=0;
```

```
      for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
      LineBuffer[i]=sum/weightsum;
     }
//---
  }
```

Prestare attenzione alla variabile i, dichiarata nella riga

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

Il suo ambito è solo per il ciclo, al di fuori di questo ciclo è un'altra variabile con lo stesso nome, dichiarata all'inizio della funzione. Inoltre, la variabile k viene dichiarata nel corpo del ciclo, il suo ambito è il corpo del ciclo.

Le variabili locali possono essere dichiarate con lo specificatore di accesso static. In questo caso, il compilatore ha una variabile nel pool globale di memoria. Pertanto, la durata di una variabile statica è pari alla durata del programma. Qui la portata di tale variabile è limitata al blocco in cui è dichiarata.

Vedi anche

Tipi di dati, Incapsulamento ed Estensibilità dei Tipi, Inizializzazione delle Variabili, Creazione ed Eliminazione di Oggetti

Creazione ed Eliminazione di oggetti

Dopo che un programma mql5 viene caricato per l'esecuzione, la memoria viene allocata per ogni variabile in base al tipo. Secondo il livello di accesso tutte le variabili sono divise in due tipi - variabili globali e variabili locali. Secondo la classe di memoria esse possono essere parametri di input di un programma mql5, static ed automatic. Se necessario, ciascuna variabile è inizializzata da un valore corrispondente. Dopo essere stata utilizzata, una variabile viene unintialized e la memoria utilizzata viene restituita al sistema eseguibile mql5.

Inizializzazione e deinizializzazione di Variabili Globali

Le variabili globali vengono inizializzate automaticamente subito dopo che un programma mql5 viene caricato e prima che qualsiasi funzione venga chiamata. Durante l'inizializzazione i valori iniziali vengono assegnati alle variabili di tipo semplice ed un costruttore (se c'è) è chiamato per gli oggetti. Le Variabili di input sono sempre dichiarate a livello globale, e vengono inizializzati da valori impostati da un utente nella finestra di dialogo durante l'avvio del programma.

Nonostante il fatto che le variabili static vengono solitamente dichiarate a livello locale, la memoria per queste variabili viene pre-assegnata, e l'inizializzazione viene eseguita subito dopo che un programma viene caricato, lo stesso per le variabili globali.

L'ordine di inizializzazione corrisponde all'ordine dichiarazione della variabile nel programma. La deinizializzazione avviene in ordine inverso. Questa regola vale solo per le variabili che non sono stati create dal nuovo operatore. Tali variabili vengono create ed inizializzate automaticamente subito dopo il caricamento, e vengono deinizializzate prima del decaricamento del programma .

Inizializzazione e Deinizializzazione di Variabili Locali

Se una variabile dichiarata a livello locale non è una static, la memoria viene allocata automaticamente per tale variabile. Le variabili locali, oltre a quelle globali, vengono inizializzate automaticamente nel momento in cui l'esecuzione del programma incontra la dichiarazione di una variabile locale. Così l'ordine di inizializzazione corrisponde all'ordine di dichiarazione.

Le variabili locali vengono deinizializzate alla fine del blocco del programma nel quale sono state dichiarate, ed in senso opposto alla loro dichiarazione. Un blocco del programma è un operatore composto che può essere una parte dell' operatore di selezione switch, l'operatore ciclico (for, while, do-while), il corpo di una funzione o una parte dell' operatore if-else.

Le variabili locali vengono inizializzate solo nel momento in cui l'esecuzione del programma incontra la dichiarazione della variabile. Se durante l'esecuzione del programma, il blocco in cui viene dichiarata la variabile non è stato eseguito, tale variabile non viene inizializzata.

Inizializzazione e Deinizializzazione di Oggetti Piazzati

Un caso particolare è quello con i puntatori ad oggetto, perché la dichiarazione di un puntatore non comporta l'inizializzazione di un oggetto corrispondente. Oggetti dinamicamente posizionati vengono inizializzati solo nel momento in cui il campione della classe viene creato dall' operatore new. L' inizializzazione di oggetti presuppone la chiamata di un costruttore di una classe corrispondente. Se non vi è alcun costruttore corrispondente nella classe, i suoi membri di un tipo semplice non verranno automaticamente inizializzati; membri dei tipi string, array dinamico ed oggetto complesso vengono inizializzati automaticamente.

I puntatori possono essere dichiarati a livello locale o globale, e possono essere inizializzati con il valore vuoto di NULL o il valore del puntatore dello stesso tipo o ereditato. Se l' operatore *new* viene chiamato per un puntatore dichiarato a livello locale, l' operatore *delete* per questo puntatore deve essere eseguito prima di uscire dal livello. In caso contrario, il puntatore verrà perso e l'eliminazione esplicita dell'oggetto avrà esito negativo.

Tutti gli oggetti creati dall'espressione di *object_pointer* =new *Class_name*, devono essere quindi eliminati per l'operatore delete(*object_pointer*). Se per qualche motivo tale variabile non viene eliminata dall' operatore delete quando il programma è completato, la voce corrispondente apparirà nel journal "Experts". Si possono dichiarare più variabili ed assegnare un puntatore di un oggetto a tutte loro.

Se un oggetto creato dinamicamente ha un costruttore, questo costruttore verrà chiamato al momento dell'esecuzione dell'operatore *new*. Se un oggetto ha un distruttore, verrà chiamato durante l'esecuzione dell' operatore *delete*.

Così oggetti dinamicamente piazzati, vengono creati solo nel momento in cui vengono creati dall'operatore *new*, e sono cancellati in modo assicurato, dall'operatore*delete* o automaticamente dal sistema di esecuzione MQL5 durante il decarico del programma. L'ordine di dichiarazione di puntatori di oggetto creato dinamicamente non influenza l'ordine di inizializzazione. L'ordine di inizializzazione e deinizializzazione è interamente controllato dal programmatore.

Allocazione dinamica della memoria in MQL5

Quando si lavora con gli array dinamici, la memoria liberata viene immediatamente restituita al sistema operativo.

Quando si lavora con oggetti di classe dinamici utilizzando l' operatore new, dapprima la memoria viene richiesta dal pool della memoria della classe con cui il gestore della memoria sta lavorando. Se non c'è abbastanza memoria nel pool, la memoria viene richiesta dal sistema operativo. Quando si elimina l'oggetto dinamico utilizzando l' l'operatore delete, la memoria liberata viene immediatamente restituica al pool di memoria della classe.

Il gestore della memoria restituisce la memoria indietro al sistema operativo immediatamente dopo l'uscita delle seguenti funzioni event handling: OnInit(), OnDeinit(), OnStart(), OnTick(), OnCalculate(), OnTimer(), OnTrade(), OnTester(), OnTesterInit(), OnTesterPass(), OnTesterDeinit(), OnChartEvent(), OnBookEvent().

Brevi caratteristiche delle Variabili

Le principali informazioni riguardo l'ordine di creazione, eliminazione, riguardo chiamate di costruttori e distruttori è riportato nella tabella che segue.

	Variabile automatica globale	Variabile automatica locale	Oggetto creato dinamicamente
Inizializzazione	subito dopo viene caricato un programma mql5	quando la linea di codice in cui viene dichiarato viene raggiunta durante l'esecuzione	all'esecuzione dell' operatore **new**

	Variabile automatica globale	Variabile automatica locale	Oggetto creato dinamicamente
Ordine di Inizializzazione	nell'ordine di dichiarazione	nell'ordine di dichiarazione	indipendentemente dall'ordine della dichiarazione
Deinizializzazione	prima che un programma mql5 viene decaricato	quando l'esecuzione esce dal blocco di dichiarazione	quando l'operatore **delete** viene eseguito o prima che un programma mql5 venga decaricato
Ordine di Deinizializzazione	nell'ordine opposto all'ordine di inizializzazione	nell'ordine opposto all'ordine di inizializzazione	indipendentemente dall'ordine di inizializzazione
Chiamata costruttore	al caricamento del programma mql5	all'inizializzazione	all'esecuzione dell' operatore *new*
Chiamata distruttore	al decaricamento del programma mql5	quando esce dal blocco in cui è stata inizializzata la variabile	all'esecuzione dell'operatore *delete*
Logs Errore	log dei messaggi sul journal "Experts" al tentativo di eliminare un oggetto creato automaticamente	log dei messaggi sul journal "Experts" al tentativo di eliminare un oggetto creato automaticamente	log dei messaggi sul journal "Experts" in merito ad oggetti dinamicamete creati non-eliminati al decarico di un programma MQL5

Vedi anche

Tipi di Dati, Incapsulamento ed Estensibilità dei Tipi, Inizializzazione delle Variabili, Ambito Visibilità e Durata delle Variabili

Preprocessore

Il preprocessore è uno speciale sottosistema del compilatore MQL5 che è inteso per la preparazione del codice sorgente del programma, immediatamente prima che il programma venga compilato.

Il preprocessore permette miglioramenti della leggibilità del codice sorgente. Il codice può essere strutturato includendo file specifici che contengono codici sorgenti di programmi-mql5. La possibilità di assegnare nomi mnemonici a costanti specifiche contribuisce al miglioramento della leggibilità del codice.

Il preprocessore permette anche la determinazione dei parametri specifici di programmi-mql5:

- Dichiara costanti
- Imposta le proprietà del programma
- Include i file in testo del programma
- Importa funzioni
- Compilazione Condizionale

Le direttive del preprocessore sono usate dal compilatore per pre-elaborare il codice sorgente prima di compilarlo. La direttiva inizia sempre con #, quindi il compilatore vieta l'utilizzo del simbolo nei nomi di variabili, funzioni, ecc.

Ogni direttiva è descritta da una voce separata ed è valida fino all'interruzione di riga. Non è possibile utilizzare più direttive in un'unica voce. Se la voce direttiva è troppo grande, può essere suddivisa in più righe usando il simbolo '\'. In questo caso, la riga successiva è considerata una continuazione della voce direttiva.

```
//+------------------------------------------------------------------+
//|  foreach pseudo-operator                                         |
//+------------------------------------------------------------------+
#define ForEach(index, array) for (int index = 0,                    \
   max_##index=ArraySize((array));                                   \
   index<max_##index; index++)
//+------------------------------------------------------------------+
//| Funzione Start programma script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   string array[]={"12","23","34","45"};
//--- bypassa l'array usando ForEach
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
/* Risultato dell'output
   0: array[0]=12
   1: array[1]=23
   2: array[2]=34
```

```
   3: array[3]=45
*/
```

Per il compilatore, tutti queste tre righe di direttive #definire sembrano un'unica lunga riga. L'esempio sopra vale anche al carattere ## che è un operatore di unione utilizzato nelle macro #define per unire i due token macro in uno solo. L'operatore di unione(merge) di token non può essere il primo o l'ultimo in una definizione di macro.

Dichiarazione di costanti (#define, #undef)

Le direttive del preprocessore sono usate dal compilatore per pre-elaborare il codice sorgente prima di compilarlo. La direttiva inizia sempre con #, quindi il compilatore vieta l'utilizzo del simbolo nei nomi di variabili, funzioni, ecc.

Ogni direttiva è descritta da una voce separata ed è valida fino all'interruzione di riga. Non è possibile utilizzare più direttive in un'unica voce. Se la voce direttiva è troppo grande, può essere suddivisa in più righe usando il simbolo '\'. In questo caso, la riga successiva è considerata una continuazione della voce direttiva.

La direttiva #define può essere utilizzata per assegnare i nomi alle costanti mnemoniche. Ci sono due forme:

```
#define definizione dell'espressione // parametro-forma libera
#define identificatore(par1,... par8) espressione    // forma parametrica
```

La direttiva #define sostitutisce l'**espressione** per tutte le altre voci trovate di *identificatore* nel testo di partenza. L'*identificatore* è sostituito solo se è un token separato. L' *identificatore* non è sostituito se è parte di un commento, parte di una stringa, o parte di un altro identificatore più.

L'identificatore costante è governato dalle stesse regole dei nomi delle variabili. Il valore può essere di qualsiasi tipo:

```
#define ABC                100
#define PI                 3.14
#define COMPANY_NAME       "MetaQuotes Software Corp."
...
void ShowCopyright()
  {
   Print("Copyright  2001-2009, ",COMPANY_NAME);
   Print("https://www.metaquotes.net");
  }
```

expression può essere costituito da vari token, quali parole chiave, costanti, espressioni costanti e non-costanti. **expression** termina con l'estremità della linea e non può essere trasferito alla linea successiva.

Esempio:

```
#define DUE         2
#define TRE       3
#define INCOMPLETO DUE+TRE
#define COMPLETO   (DUE+TRE)
voidOnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETO*2);
   Print("(2 + 3)*2 = ",COMPLETO*2);
  }
// Risultato
// 2 + 3*2 = 8
// (2 + 3)*2 = 10
```

Forma parametrica #define

With the parametric form, all the subsequent found entries of identifier will be replaced by expression taking into account the actual parameters. Ad esempio:

```
 // esempio con due paremetri, a e b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
l'esprezzione double c=MUL(A,B);
è equivalente a double c=((2+3)*(5-1));
*/
// Risultato
// c=20
```

Accertarsi di racchiudere i parametri tra parentesi quando si utilizzano i parametri di espressione, in modo da evitare errori non-ovvi che sono difficili da trovare. Se si riscrive il codice senza usare le parentesi, il risultato sarà diverso:

```
 // esempio con due paremetri, a e b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
l'esprezzione double c=MUL(A,B);
è equivalente a double c=2+3*5-1;
*/
// Risultato
// c=16
```

Quando si utilizza la forma parametrica, un massimo di 8 parametri sono ammessi.

```
 // forma parametrica corretta
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // un parametro - 'text'

 // forma parametrica errata
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // più di 8 parame
```

The #undef directive

The #undef directive cancels declaration of the macro substitution, defined before.

Esempio:

```
#define MACRO

void func1()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* Risultato:
 MACRO is defined in func1
 MACRO is not defined in func2
*/
```

Vedi anche

Identificatori, Costanti Carattere

Proprietà del programma (#property)

Ogni programma MQL5 permette di specificare ulteriori parametri specifici denominati #property che aiutano il terminale client nella corretta revisione dei programmi senza la necessità di avviarli in modo esplicito. Ciò riguarda prima di tuttto le impostazioni degli indicatori esterni, le proprietà descritte nei file inclusi vengono completamente ignorate. Le proprietà devono essere specificate nel file mq5 principale.

```
#property valore dell'identificatore
```

Il compilatore scriverà valori dichiarati nella configurazione del modulo eseguito.

Costante	Tipo	Descrizione
icon	string	Il percorso al file di un'immagine che verrà utilizzata come icona del programma EX5. Le regole specifiche del percorso sono le stesse per risorse. La proprietà deve essere specificata nel modulo principale con il codice sorgente MQL5. Il file deve essere in formato ICO.
link	string	Link al sito web della società
copyright	string	Il nome della società
version	string	Versione Programma, massimo 31 caratteri
descrizione	string	Breve descrizione del testo di un programma mql5. Possono essere presenti alcune delle *descrizioni*, ciascuna delle quali descrive una riga di testo. La lunghezza totale di tutte le *descrizioni* non può superare i 511 caratteri compresi gli avanzamento riga.
stacksize	int	Grandezza pila del programma MQL5. La pila(stack) di grandezza sufficiente è necessaria quando si eseguono chiamate ricorsive. Quando si avvia uno script o un Expert Advisor sul chart, viene allocata una pila di almeno 8 MB. In caso di indicatori, la dimensione della pila è sempre fissa ed uguale a 1 MB. Quando un programma viene lanciato nello strategy tester, lo stack di 16 MB viene sempre assegnato per esso.
library		Una libreria, nessuna funzione start è assegnata; funzioni con il modificatore export possono essere importate in altri programmi MQL5
indicator_applied_price	int	Specifica il valore predefinito per il campo "Applica a". È possibile specificare uno dei valori di ENUM_APPLIED_PRICE. Se la proprietà non è specificata, il valore predefinito è PRICE_CLOSE

Costante	Tipo	Descrizione
indicator_chart_window		Mostra l'indicatore nella finestra del chart
indicator_separate_window		Mostra l'indicatore in una finestra separata
indicator_height	int	Altezza fissa della sottofinestra indicatore in pixel (proprietà INDICATOR_HEIGHT)
indicator_buffers	int	Numero di buffer per il calcolo dell'indicatore
indicator_plots	int	Numero di serie grafiche nell'indicatore
indicator_minimum	double	Il limite inferiore di scala per una finestra indicatore separata
indicator_maximum	double	Il limite superiore di scala per una finestra di indicatore separata
indicator_labelN	string	Consente di impostare una etichetta per la N-esima serie grafica visualizzata in DataWindow. Per le serie grafiche che richiedono buffer indicatori multipli (DRAW_CANDLES, DRAW_FILLING e altri), i nomi delle variabili sono definiti dal separatore ';'.
indicator_colorN	color	Il colore per visualizzare la linea N, dove N è il numero di serie grafiche; la numerazione parte da 1
indicator_widthN	int	Spessore linea nelle serie grafiche, dove la numerazione di N - numero di serie grafica, inizia da 1
indicator_styleN	int	Stile di linea nelle serie grafiche, specificato dai valori di ENUM_LINE_STYLE. N - numero di serie grafiche, la numerazione inizia da 1
indicator_typeN	int	Tipo di grafico tracciato, indicato dai valori di ENUM_DRAW_TYPE. N - numero di serie grafica, la numerazione inizia da 1
indicator_levelN	double	Livello orizzontale di N in una finestra di indicazione separata
indicator_levelcolor	color	Colore di livelli orizzontali dell'indicatore
indicator_levelwidth	int	Spessore di livelli orizzontali dell'indicatore
indicator_levelstyle	int	Stile di livelli orizzontali dell'indicatore
script_show_confirm		Visualizzazione di una finestra di conferma prima di eseguire lo script
script_show_inputs		Visualizzazione di una finestra con le proprietà prima di eseguire lo script e disattivazione di questa finestra di conferma
tester_indicator	string	Il nome di un indicatore personalizzato nel formato "*indicator_name.ex5*". Indicatori che richiedono

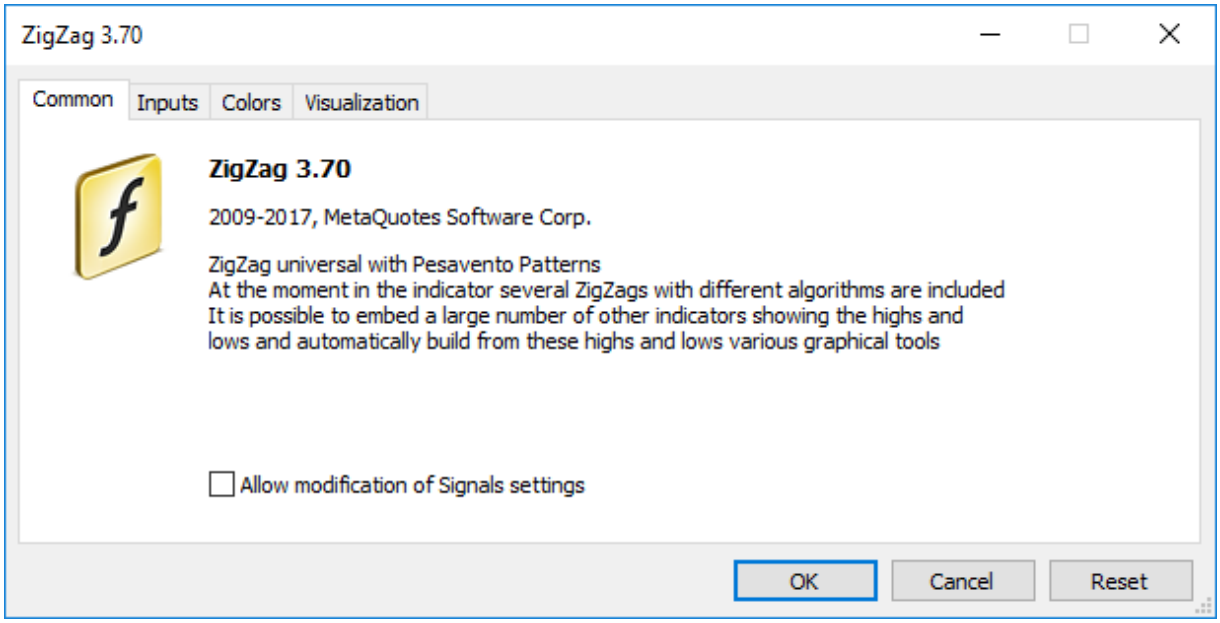
Costante	Tipo	Descrizione
		testing sono definiti automaticamente dalla chiamata della funzione iCustom(), se il parametro corrispondente viene impostato tramite una costante stringa. Per tutti gli altri casi (utilizzo della funzione IndicatorCreate() o uso di una non costante stringa nel parametro che imposta il nome indicatore) questa proprietà è obbligatoria
tester_file	string	Nome file per un tester con l'indicazione di estensione, tra virgolette (come una costante stringa). Il file specificato verrà passato al tester. File di input da testare, se ci sono quelli necessari, deve sempre essere specificato.
tester_library	string	Nome libreria con l'estensione, tra virgolette. Una libreria può avere estensione DLL o EX5. Librerie che richiedono il testing vengono definite automaticamente. Tuttavia, se una delle librerie viene utilizzata da un indicatore personalizzato, questa proprietà è necessaria.
tester_set	string	Nome del file impostato con i valori e lo steo dei parametri di input. Il file viene passato al tester prima del test e dell'ottimizzazione. Il nome del file viene specificato con un'estensione e le doppie virgolette come una costante stringa. Se si specifica il nome EA e il numero di versione come "<nome_expert> _ <numero> .set " in un nome file impostato, allora viene automaticamente aggiunto al menu di download delle versioni dei parametri sotto numero della versione <numero> . Ad esempio, il nome "MACD Sample_4.set" indica che si tratta di un file impostato per l'EA "MACD Sample.mq5" con il numero di versione uguale a 4. Per studiare il formato, si consiglia di salvare manualmente le impostazioni di test/ottimizzazione nel tester strategia e quindi aprire il file set creato in questo modo.
tester_no_cache	string	Durante l'esecuzione dell'ottimizzazione, il tester della strategia salva tutti i risultati dei passaggi eseguiti su cache di ottimizzazione, in cui il risultato del test viene salvato per ogni set di parametri di input. Ciò consente di utilizzare i risultati predefiniti durante la ri-ottimizzazione sugli stessi parametri senza perdite di tempo nel ricalcolo.

Costante	Tipo	Descrizione
		Ma in alcune attività (ad esempio nei calcoli matematici), potrebbe essere necessario eseguire calcoli indipendentemente dalla disponibilità di risultati-pronti nella cache di ottimizzazione. In questo caso, il file dovrebbe includere la proprietà *tester_no_cache*. I risultati del test sono sempre memorizzati in cache, in modo da poter visualizzare tutti i dati sui passaggi eseguiti nel tester di strategia.
tester_everytick_calculate	string	Nelno Strategy Tester, gli indicatori vengono calcolati solo quando si accede ai loro dati, ovvero quando vengono richiesti i valori dei buffer degli indicatori. Ciò fornisce una velocità di test e ottimizzazione significativamente più veloce, se non è necessario ottenere valori di indicatori su ogni tick. Specificando la proprietà *tester_everytick_calculate*, è possibile attivare il calcolo forzato dell'indicatore su ogni tick. Gli indicatori nel Tester di Strategia sono anche calcolati forzatamente su ogni tick, nei seguenti casi: • durante il test nella modalità visiva; • se l'indicatore ha una delle seguenti funzioni: EventChartCustom, OnChartEvent, OnTimer; • se l'indicatore è stato creato usando il compilatore con numero di build sotto il 1916. Questa funzione si applica solo allo Strategy Tester, mentre negli indicatori del terminale vengono sempre calcolati su ogni tick ricevuto.
optimization_chart_mode	string	Specifica il tipo di chart ed i nomi di due parametri di input che verranno utilizzati per la visualizzazione dei risultati di ottimizzazione. Ad esempio, "3d, InpX, InpY" significa che i risultati saranno mostrati in un grafico 3D con gli assi delle coordinate basati sui valori dei parametri InpX e InpY testati. Pertanto, la proprietà abilita la specifica dei parametri che verranno utilizzati per visualizzare il grafico di ottimizzazione ed il tipo di grafico, direttamente nel codice del programma. Possibili opzioni:

Costante	Tipo	Descrizione
		• "3d, input_parameter_name1, input_parameter_name2" significa un chart di visualizzazione 3D, che può essere ruotato, ingrandito e ridotto. Il chart viene creato utilizzando due parametri. • "2d, input_parameter_name1, input_parameter_name2" indica un chart a griglia 2D, in cui ogni cella viene disegnata in un determinato colore a seconda del risultato. Il chart viene creato utilizzando due parametri. • "1d, input_parameter_name1, input_parameter_name2" significa un chart lineare, in cui i risultati sono ordinati in base al parametro specificato. Ogni passaggio viene visualizzato come un punto. Il chart è basato su un parametro. • "0d, input_parameter_name1, input_parameter_name2" indica un chart normale con risultati ordinati in base all'ora di arrivo del risultato del passaggio. Ogni passaggio viene visualizzato come un punto nel chart. L'indicazione dei parametri non è richiesta, ma i parametri specificati possono essere utilizzati per il passaggio manuale ad altri tipi di chart. Facoltativamente, è possibile indicare solo il tipo di chart, senza specificare uno o due parametri di input. In questo caso, il terminale selezionerà i parametri richiesti per mostrare il chart di ottimizzazione.

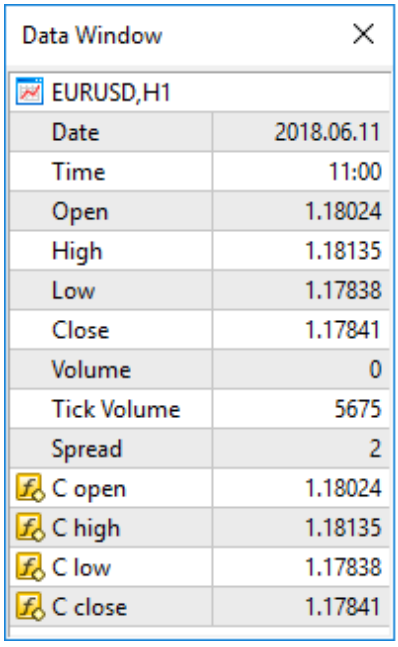
Esempio di Task di Descrizione e Numero di Versione

```
#property version     "3.70"      // Versione corrente dell' Expert Advisor
#property description "ZigZag universal con Patterns Pesavento"
#property description "Al momento sono inclusi nell'indicatore vari Zig-Zag con divers
#propertydescription"E' possibile inserire un gran numero di altri indicatori che most
#propertydescription"bassi e generare automaticamente da questi alti e bassi, vari sti
```



Esempi di Specificare un' Etichetta Separata per ogni Indicatore Buffer ("C open; C high; C low; C close")

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots   1
#property indicator_type1   DRAW_CANDLES
#property indicator_width1  3
#property indicator_label1  "C open;C high;C low;C close"
```



Includere Files (# include)

La riga di comando *#include* può essere posizionata in qualsiasi punto del programma, ma di solito tutte le inclusioni sono poste all'inizio del codice sorgente. Formato Chiamata:

```
#include <file_name>
#include "file_name"
```

Esempi:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

Il preprocessore rimpiazza la riga *#include* <*file_name*> con il contenuto del file WinUser32.mqh. Le parentesi angolari indicano che il file WinUser32.mqh sarà preso dalla directory standard (di solito è *terminal_installation_directory**MQL5**Include*). La directory corrente non è visibile.

Se il nome del file è racchiuso tra virgolette, la ricerca viene effettuata nella directory corrente (che contiene il file sorgente principale). La directory standard non è visibile.

Vedi anche

Standard Library, Importazione di Funzioni

Funzione di importazione (#import)

Le funzioni sono importate da moduli compilati MQL5 (files *.ex5) e dai moduli del sistema operativo (files *.dll). Il nome del modulo è specificato nella direttiva *#import*. Per far si che il compilatore sia in grado di formare correttamente la chiamata della funzione importata ed organizzare la corretta trasmissione dei parametri, la descrizione completa delle funzioni è necessaria. Descrizioni delle funzioni immediatamente dopo la direttiva *#import "nome modulo"*. Nuovo comando *#import* (può essere senza parametri) completa il blocco di descrizioni delle funzioni importate.

```
#import "file_name"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

Funzioni importate possono avere qualsiasi nome. Funzioni aventi gli stessi nomi, ma da diversi moduli possono essere importate contemporaneamente. Funzioni importate possono avere nomi che coincidono con i nomi delle funzioni incorporate. L' operazione di risoluzione ambito definisce quale delle funzioni dovrebbe essere chiamata.

L'ordine di ricerca di un file specificato dopo la parola chiave #import è descritto in Chiamata delle Funzioni Importate.

Poiché le funzioni importate sono al di fuori del modulo compilato, il compilatore non è in grado di verificare la validità dei parametri passati. Pertanto, per evitare errori di run-time, si deve descrivere con precisione la composizione e l'ordine dei parametri passati alle funzioni importate. I parametri passati alle funzioni importate (sia da EX5, e dal modulo DLL) possono avere valori di default.

I seguenti non possono essere utilizzati per i parametri di funzioni importate:

- puntatori (*);
- collegamenti ad oggetti che contengono array dinamici e/o puntatori.

Le classi, array di stringhe o oggetti complessi che contengono stringhe e/o array dinamici di ogni tipo non possono essere passati come parametro alle funzioni importate dalla DLL.

Esempi:

```
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int valore_rosso,int valore_verde,int valore_blu);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double numero,int precisione);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
```

```
#import
```

Per importare le funzioni durante l'esecuzione di un programma di MQL5, viene utilizzata l'associazione anticipata. Ciò significa che la libreria viene caricata durante il caricamento di un programma con il suo programma EX5.

Non è raccomandato l'uso di un nome completo qualificato del modulo caricabile del tipo *Unità:\Cartella\NomeFile.Ext*. Le librerie MQL5 vengono caricate dalla cartella *terminal_dir\MQL5\Libraries*.

Se la funzione importata ha versioni di chiamata diverse per le versioni Windows a 32 e 64 bit, entrambe dovrebbero essere importate, e la versione corretta della funzione dovrebbe essere chiamata esplicitamente usando la variabile _IsX64.

Esempio:

```
#import "user32.dll"
//--- Per il sistema a 32 bit
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- Per il sistema a 64 bit
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//| MessageBox_32_64_bit utilizza la versione corretta di MessageBoxW|
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   //--- Se stiamo usando Windows a 64 bit
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"64-bit MessageBoxW call example","MessageBoxW 64 bit",MB_C
     }
   else  // Stiamo usando Windows a 32-bit
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"32-bit MessageBoxW esempio di chiamata","MessageBoxW 32 bi
     }
   return (res);
  }
//+------------------------------------------------------------------+
//| Funzione Start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
   PrintFormat("MessageBox_32_64_bit ha restituito %d",ans);
  }
```

Importazione di funzioni da librerie .NET

Per lavorare con le funzioni di libreria .NET, è sufficiente importare la DLL stessa senza definire funzioni specifiche. MetaEditor importa automaticamente tutte le funzioni con cui è possibile lavorare:

- Strutture semplici (POD, semplici vecchi dati) — strutture che contengono solo tipi di dati semplici.
- Funzioni statiche pubbliche aventi parametri in cui vengono utilizzati solo tipi semplici e strutture POD o i loro array

Per chiamare le funzioni dalla libreria, è sufficiente importarle:

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

Il codice C# della funzione Inc di TestClass ha il seguente aspetto:

```
public class TestClass
  {
   public static void Inc(ref int x)
      {
       x++;
      }
  }
```

Come risultato dell'esecuzione, lo script restituisce il valore di 42.

Vedi anche

Inclusione Files

Compilazione condizionale (#ifdef, #ifndef, #else, #endif)

Le direttive del preprocessore sono usate dal compilatore per pre-elaborare il codice sorgente prima di compilarlo. La direttiva inizia sempre con #, quindi il compilatore vieta l'utilizzo del simbolo nei nomi di variabili, funzioni, ecc.

Ogni direttiva è descritta da una voce separata ed è valida fino all'interruzione di riga. Non è possibile utilizzare più direttive in un'unica voce. Se la voce direttiva è troppo grande, può essere suddivisa in più righe usando il simbolo '\'. In questo caso, la riga successiva è considerata una continuazione della voce direttiva.

Direttive di compilazione condizionale del preprocessore permettono la compilazione o il salto di una parte del programma a seconda della realizzazione di una certa condizione.

Tale condizione può assumere una delle seguenti forme.

```
#ifdef identifier
   // il codice che si trova qui viene compilato se l'identificatore è già stato defi
#endif
```

```
#ifndef identifier
   // il codice che si trova qui viene compilato se l'identificatore non è attualmente
#endif
```

Qualsiasi direttiva di compilazione condizionale può essere seguita da un numero qualsiasi di righe eventualmente contenenti la direttiva #else e terminante con #endif. Se la condizione verificata è vera, le righe tra #else ed #endif vengono ignorate. Se la condizione verificata non è soddisfatta, tutte le righe tra il controllo e la direttiva #else (o direttiva #endif se la prima è assente) vengono ignorate.

Esempio:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   #ifdef TestMode
      Print("Modalità Test");
   #else
      Print("Modalità Normale");
   #endif
  }
```

A seconda del tipo di programma e modalità di compilazione, le macro standard sono definite nel seguente modo:

la macro __MQL5__ viene definita quando si compila il file *.mq5, la macro __ MQL4__ viene definita durante la compilazione di un *.mq4.

La macro _DEBUG viene definita durante la compilazione in modalità di debug.
La macro _RELEASE viene definita durante la compilazione in modalità di rilascio.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   #ifdef __MQL5__
      #ifdef _DEBUG
         Print("Saluti dal compilatore MQL5 [DEBUG]");
      #else
        #ifdef _RELEASE
           Print("Saluti dal compilatore MQL5 [RELEASE]");
        #endif
     #endif
   #else
      #ifdef __MQL4__
         #ifdef _DEBUG
           Print("Saluti dal compilatore MQL4 [DEBUG]");
        #else
           #ifdef _RELEASE
              Print("Saluti dal compilatore MQL4 [RELEASE]");
           #endif
        #endif
     #endif
   #endif
  }
```

Programmazione Ad Oggetti (OOP Object-Oriented-Programming)

La Programmazione Orientata agli Oggetti (OOP) è la programmazione focalizzata primariamente sui dati, mentre i dati ed il comportamento sono stati indissolubilmente legati. Dati e comportamento insieme costituiscono una classe, mentre gli oggetti sono istanze della classe.

I componenti dell' approccio object-oriented sono:

- Incapsulamento ed estensibilità del tipo
- Ereditarietà
- Polimorfismo
- Sovraccarico
- Funzioni virtuali

L' OOP considera il calcolo come la modellazione del comportamento. L'elemento modellato è l'oggetto rappresentato da astrazioni computazionali. Supponiamo di voler scrivere il noto gioco "Tetris". Per fare questo, dobbiamo imparare a modellare l'apparenza di forme casuali composte da quattro quadrati uniti tra loro da bordi. Inoltre abbiamo bisogno di regolare la velocità di caduta delle forme, definire le operazioni di rotazione e lo spostamento delle forme. Lo spostamento delle forme sullo schermo è limitato da confini del 'pozzo', tale requisito deve anch'essere modellato. Oltre a ciò, le righe riempite di cubi devono essere distrutte ed i punti conseguiti devono essere contati.

Così, questo gioco di-facile-comprensione richiede la creazione di vari modelli - modello delle forme, modello del pozzo, il modello del movimento della forma e così via. Tutti questi modelli sono astrazioni, rappresentate dai calcoli del computer. Per descrivere questi modelli, viene usato il concetto di Tipo di Dati Astratti, ADT (o tipo di dati complessi). Strettamente parlando, il modello del movimento delle "forme" nel DOM non è un tipo di dati, ma è un insieme di operazioni sul tipo di dati "forma", utilizzando le restrizioni del tipo di dati "pozzo".

Gli oggetti sono variabili classe. La programmazione orientata agli oggetti permette di creare e utilizzare facilmente ADT. La programmazione orientata agli oggetti utilizza il meccanismo dell' ereditarietà. Il vantaggio dell' ereditarietà è nel fatto che essa consente di ottenere tipi derivati da tipi di dati già definiti dall'utente.

Ad esempio, per creare le forme del Tetris, è conveniente creare prima una classe base Shape. Le altre classi che rappresentano tutti i tipi di sette forme possibili si può ottenere sulla sua base. Il comportamento delle forme è definito nella classe base, mentre l'attuazione del comportamento di ogni forma separata è definito nelle classi derivate.

Nella programmazione orientata agli oggetti, essi sono responsabili per il loro comportamento. Lo sviluppatore ADT dovrebbe includere un codice per descrivere qualsiasi comportamento che normalmente ci si aspetterebbe dagli oggetti corrispondenti. Il fatto che l'oggetto stesso è responsabile per il suo comportamento, semplifica notevolmente il compito di programmazione per l'utente di questo oggetto.

Se vogliamo disegnare una forma sullo schermo, abbiamo bisogno di sapere dove sarà il centro e come disegnarlo. Se una forma separata sa come disegnare se stessa, il programmatore deve inviare un messaggio di "disegna" quando si utilizza tale forma.

Il linguaggio MQL5 è come il C++ , e ha anche il meccanismo dell' incapsulamento per l'implementazione dell' ADT. Da un lato l'incapsulamento combina i dettagli interni di attuazione di un particolare tipo, e dall'altro combina funzioni accessibili esternamente che possono influenzare oggetti di questo tipo. I dettagli dell'implementazione possono essere inaccessibili per un programma che utilizza questo tipo.

Il concetto di OOP ha un insieme di concetti, tra cui:

- Simulazione di azioni dal mondo reale
- Tipi di dato, dall'utente definiti
- Nascondere i dettagli di implementazione
- Possibilità di riutilizzo del codice attraverso l'ereditarietà
- Interpretazione delle chiamate di funzione durante l'esecuzione

Alcuni di questi concetti sono piuttosto vaghi, alcuni sono astratti, altri sono generali.

Incapsulamento ed estensibilità dei Tipi

OOP è un approccio equilibrato alla scrittura di software. I dati e il comportamento sono impacchettati insieme. L'incapsulamento crea tipi di dati definiti dall'utente, estendendo i tipi di dati del linguaggio ed interagendo con essi. L' estensibilità dei tipi è la possibilità di aggiungere al linguaggio tipi di dati definiti-dall'utente, che sono anche facili da utilizzare, così come i tipi di base.

Un tipo di dato astratto, per esempio, una stringa, è una descrizione dell' ideale, ben noto, comportamento del tipo.

L'utente che utlizza la stringa sa che le operazioni sulle stringhe, quali concatenazione o stampa, hanno un certo comportamento. Operazioni di concatenazione e stampa sono chiamate metodi.

Una certa implementazione di ADT può avere alcune limitazioni, per esempio, le stringhe possono essere limitate in lunghezza. Queste limitazioni influenzano il comportamento aperto a tutte. Al tempo stesso, dettagli di implementazione interni o privati non influenzano direttamente il modo in cui l'utente vede l'oggetto. Ad esempio, la stringa è spesso implementata come un array, mentre l'indirizzo di base interno di questo array ed il suo nome non sono essenziali per l'utente.

L'incapsulamento è la capacità di nascondere i dettagli di implementazione quando vengono fornite le interfacce aperte per il tipo definito-dall'utente. In MQL5, così come in C++, definizioni di classe e struttura (class e struct) vengono utilizzate per le disposizioni di incapsulamento in combinazione con le parole chiave di accesso private, protected e public.

La parola chiave public indica che l'accesso ai membri che stanno dietro di esso, è aperto senza restrizioni. Senza questa parola chiave, i membri della classe sono bloccati per impostazione predefinita. I membri privati sono accessibili solo da funzioni che sono membro soltanto della sua classe.

Funzioni di classe protette sono disponibili per funzioni di classe non solo nella sua classe, ma anche nelle sue classi eredi. Funzioni di classe pubbliche sono disponibili per qualsiasi funzione nell'ambito della dichiarazione di classe. La protezione permette di nascondere parte dell'implementazione della classe, impedendo cambiamenti inattesi nella struttura dei dati. Restrizioni all' accesso oppure oscuramento dei dati sono una caratteristica della programmazione orientata agli oggetti.

Di solito, le funzioni di classe sono protette e dichiarate con il modificatore protected, la lettura e la scrittura dei valori vengono eseguite utilizzando cosiddetti speciali metodi set-and-get che sono definiti dai modificatori pubblici di accesso.

Esempio:

```
class CPerson
  {
protected:
   string            m_name;                    // name
public:
   void              SetName(string n){m_name=n;}// imposta il nome
   string            GetName(){return (m_name);} // restituisce il nome
  };
```

Questo approccio offre diversi vantaggi. In primo luogo, dal nome della funzione possiamo capire ciò che fa - imposta o ottiene il valore di un membro della classe. In secondo luogo, forse in futuro ci sarà bisogno di cambiare il tipo della variabile m_name nella classe CPerson o in una delle sue classi derivate.

In questo caso, abbiamo bisogno solo di modificare l'implementazione delle funzioni SetName() e GetName(), mentre gli oggetti della classe CPerson saranno disponibili per l'utilizzo in un programma senza modifiche al codice, perché l'utente non sa nemmeno che il tipo di dati di m_name è cambiato.

Esempio:

```
struct Name
  {
   string            first_name;                  // nome
   string            last_name;                   // cognome
  };

class CPerson
  {
protected:
   Name              m_name;                      // nome
public:
   void              SetName(string n);
   string            GetName(){return(m_name.first_name+" "+m_name.last_name);}
private:
   string            GetFirstName(string full_name);
   string            GetLastName(string full_name);
  };

void CPerson::SetName(string n)
  {
   m_name.first_name=GetFirstName(n);
   m_name.last_name=GetLastName(n);
  }

string CPerson::GetFirstName(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>0) StringSetCharacter(full_name,pos,0);
   return(full_name);
  }

string CPerson::GetLastName(string full_name)
  {
   string ret_string;
   int pos=StringFind(full_name," ");
   if(pos>0) ret_string=StringSubstr(full_name,pos+1);
   else      ret_string=full_name;
   return(ret_string);
  }
```

Vedi anche

Tipi di Dati

Ereditarietà

La caratteristica della OOP è l'incoraggiamento del riutilizzo del codice attraverso l'ereditarietà. Una nuova classe viene costituita da una esistente, che è chiamata la classe base. La classe derivata utilizza i membri della classe base, ma può anche modificarli e completarli.

Molti tipi sono variazioni dei tipi esistenti. È spesso tediante sviluppare un nuovo codice per ciascuno di essi. Inoltre, un nuovo codice comporta nuovi errori. La classe derivata eredita la descrizione della classe base, in tal modo qualsiasi nuovo sviluppo e re-testing del codice non è necessario. Le relazioni di ereditarietà sono gerarchiche.

La gerarchia è un metodo che permette di copiare gli elementi in tutta la loro diversità e complessità. Esso introduce la classificazione oggetti. Ad esempio, la tavola periodica degli elementi ha i gas. Essi possiedono le proprietà intrinseche a tutti gli elementi periodici.

Gas inerti costituiscono la successiva sottoclasse più importante. La gerarchia è che il gas inerte, come l'argon, è un gas, ed il gas, a sua volta, è parte del sistema. Tale gerarchia permette di interpretare il comportamento di gas inerti facilmente. Sappiamo che i loro atomi contengono protoni ed elettroni, che è vero per tutti gli altri elementi.

Sappiamo che sono in uno stato gassoso a temperatura ambiente, come tutti i gas. Sappiamo che nessun gas della sottoclasse dei gas inerti, entra usualmente in reazione chimica con altri elementi, ed è una proprietà di tutti i gas inerti.

Si consideri un esempio della successione di forme geometriche. Per descrivere la varietà di forme semplici (cerchio, triangolo, rettangolo, quadrato, ecc.), il modo migliore è quello di creare una classe base (ADT), che è l'antenato di tutte le classi derivate.

Creiamo un classe base CShape, che contiene solo i componenti più comuni che descrivono la forma. Questi membri descrivono proprietà caratteristiche a qualsiasi forma - il tipo di forma e le principali coordinate di punto di ancoraggio.

Esempio:

```
//--- La classe base Shape
class CShape
  {
protected:
   int       m_type;                    // tipo di Shape(forma)
   int       m_xpos;                    // X - coordinate del punto base
   int       m_ypos;                    // Y - coordinate del punto base
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // costruttore
   void      SetXPos(int x){m_xpos=x;} // imposto X
   void      SetYPos(int y){m_ypos=y;} // imposto Y
  };
```

Successivamente, creare nuove classi derivate dalla classe base, in cui si andranno ad aggiungere i campi necessari, ognuna delle quali specifica una certa classe. Per la forma del Cerchio è necessario aggiungere un membro che contiene il valore del raggio. La forma del Quadrato è caratterizzata dal valore del lato. Pertanto, classi derivate, ereditate dalla classe base CShape verranno dichiarate come segue:

```
//--- La classe derivata cerchio
class CCircle : public CShape          // Dopo il due_punti definiamo la classe base
  {                                    // dalla quale viene fatta l'eredità
private:
   int             m_radius;           // raggio del cerchio

public:
                   CCircle(){m_type=1;}// costruttore, type 1
  };
```

La dichiarazione della classe della forma del Quadrato, è simile:

```
//--- la classe derivata Square
class CSquare : public CShape        // Dopo il due_punti definiamo la classe base
  {                                    // dalla quale viene fatta l'eredità
private:
   int             m_square_side;      // lato del quadrato

public:
                   CSquare(){m_type=2;} // costruttore, type 2
  };
```

Occorre notare che mentre l'oggetto viene creato viene chiamato prima il costruttore della classe base, e poi viene chiamato il costruttore della classe derivata. Quando un oggetto viene distrutto viene prima chiamato il distruttore della classe derivata, e poi viene chiamato il distruttore della classe base.

Così, dichiarando i membri più generici nella classe base, possiamo aggiungere i membri addizionali nelle classi derivate, che specificano una classe particolare. L'ereditarietà consente di creare potenti librerie di codice che possono essere riutilizzate più volte.

La sintassi per creare una classe derivata da una già esistente è la seguente:

```
class class_name :
          (public | protected | private) opt  base_class_name
  {
   dichiarazione di membri della classe
  };
```

Uno degli aspetti della classe derivata è la visibilità (apertura) dei suoi membri successori (eredi). Le parole chiave public, protected e private vengono utilizzate per indicare il grado, con cui i membri della classe base saranno disponibili per qualla derivata. La parola chiave public dopo i due punti nell'intestazione di una classe derivata indica che i membri protetti e pubblici della classe base CShape devono essere ereditati come membri protetti e pubblici della classe derivata CCircle.

I membri della classe private della classe base non sono disponibili per la classe derivata. L'ereditarietà public anche significa che le classi derivate (CCircle e CSquare) sono CShapes. Cioè, il Quadrato (CSquare) è una forma (CShape), ma la forma non deve necessariamente essere un quadrato.

La classe derivata è una modifica della classe base, essa eredita i membri protetti e pubblici della classe base. I costruttori e distruttori della classe di base non possono essere ereditati. Oltre ai membri della classe base, nuovi membri vengono aggiunti nella classe derivata.

La classe derivata può includere l'implementazione di funzioni membro, diverse dalla classe base. Non ha niente in comune con l' overload, laddove il significato del nome della stessa funzione può essere diverso per le differenti sigle.

Nell' ereditarietà protected, i membri public e protected della classe base divengono membri protected della classe derivata. Nell' ereditarietà private, i membri public e protected della classe base divengono membri private della classe derivata.

Nelle ereditarietà protected e private, la relazione che "l'oggetto di una classe derivata sia l' oggetto di una classe base" non è vero. Le ereditarietà protected e private sono rare, e ciascuna di esse deve essere usata con attenzione.

Si dovrebbe comprendere che il tipo di ereditarietà (public, protected o private) non influenza le vie di **accesso ai membri delle classi base nella gerarchia di ereditarietà da una classe derivata**. Con qualunque tipo di eredità, solo i membri della classe base dichiarata con gli specificatori di accesso public e protected saranno disponibili fuori dalle classi derivate. Prendiamolo in considerazione nel seguente esempio:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------------------+
//| Esempio di classe con un qualche tipo di accesso                             |
//+------------------------------------------------------------------------------+
class CBaseClass
  {
private:              //--- Il membro private non è disponibile da classi derivate
   int                m_member;
protected:            //--- Il metodo protected è disponibile dalla classe base e dalle
   int                Member(){return(m_member);}
public:               //--- Il costruttore della classe è disponibile per tutti i membr
                      CBaseClass(){m_member=5;return;};
private:              //--- Un metodo privato per l'assegnazione di un valore ad m_memb
   void               Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------------------+
//| Classe derivata con errori                                                   |
//+------------------------------------------------------------------------------+
class CDerived: public CBaseClass // la specifica di ereditarietà pubblica può essere
  {
public:
   void Func() // Nella classe derivata, definire una funzione con le chiamate ai memb
     {
      //--- Un tentativo di modifica di un membro private della classe base
      m_member=0;         // Errore, il membro private della classe base non è disponib
```

```
      Member(0);          // Errore, il metodo private della classe base non è disponi
      //--- Lettura del membro della classe base
      Print(m_member);    // Errore, il membro private della classe base non è disponi
      Print(Member());    // Nessun errore, il metodo di protezione è disponibile dall
     }
  };
```

Nell'esempio precedente, CBaseClass ha solo il metodo public - il costruttore. I costruttori vengono chiamati automaticamente quando si crea un oggetto di classe. Pertanto, il membro private m_member ed i medoti protected Member() non possono essere chiamati dall'esterno. Ma in caso di ereditarietà public, il metodo Member() della classe base sarà disponibile dalle classi derivate.

In caso di ereditarietà protected, tutti i membri della classe base con accesso public e protected diventano protected. Ciò significa che se i dati public dei membri e metodi della classe di base erano accessibili dall'esterno, con l'ereditarietà protected sono disponibili solo dalle classi della classe derivata e dei suoi ulteriori derivati.

```
//+------------------------------------------------------------------------------+
//| Esempio di classe con un qualche tipo di accesso                             |
//+------------------------------------------------------------------------------+
class CBaseMathClass
  {
private:              //--- Il membro private non è disponibile da classi derivate
   double             m_Pi;
public:               //--- Ottenere ed impostare un valore per m_Pi
   void               SetPI(double v){m_Pi=v;return;};
   double             GetPI(){return m_Pi;};
public:               // Il costruttore della classe è a disposizione di tutti i membri
                      CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------------------+
//| Classe derivata, in cui m_Pi non può essere modificato                       |
//+------------------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // Ereditarietà protected
  {
private:
   double             m_radius;
public:               //--- Metodi public nella classe derivata
   void               SetRadius(double r){m_radius=r; return;};
   double             GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------------------+
//| Funzione di starting dello Script                                            |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Quando si crea una classe derivata, il costruttore della classe base verrà chiam
   CProtectedChildClass pt;
//--- Specificare il raggio
   pt.SetRadius(10);
```

```
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- Se commentiamo la stringa qui di seguito, verrà visualizzato un errore in fase c
// pt.SetPI(3);

//--- Ora dichiariamo una variabile della classe base e cerchiamo di impostare la cost
   CBaseMathClass bc;
   bc.SetPI(10);
//--- Ecco il risultato
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

L'esempio mostra che i metodi SetPi() e GetPi() nella classe base CBaseMathClass sono aperti e disponibili per essere chiamati da qualsiasi punto del programma. Ma al tempo stesso, per CProtectedChildClass che è derivato da esso questi metodi possono essere chiamati solo dai metodi della classe CProtectedChildClass o sue classi derivate.

In caso di ereditarietà private, tutti i membri della classe di base con accesso public e protected diventato private, e chiamarli diventa impossibile in un' ulteriore eredità.

MQL5 non ha l'ereditarietà multipla.

Vedi anche

Strutture e Classi

Polimorfismo

Il polimorfismo è una opportunità per le diverse classi di oggetti, relazionate attraverso l'ereditarietà, per rispondere in vari modi quando si chiama lo stesso elemento della funzione. Contribuisce a creare un meccanismo universale che descrive il comportamento non solo della classe base, ma anche delle classi discendenti.

Continuiamo a sviluppare una classe base CShape ,e definiamo un membro della funzione GetArea(), designato per calcolare l'area di una forma. In tutte le classi discendenti, prodotto per eredità dalla classe base, ri-definiamo questa funzione secondo le regole del calcolo dell'area di una forma particolare.

Per un quadrato (classe CSquare), l'area è calcolata attraverso i suoi lati, per un cerchio (classe CCircle), l'area si esprime attraverso il suo raggio, ecc. Siamo in grado di creare un array per memorizzare gli oggetti di tipo CShape, in cui entrambi gli oggetti di una classe base e quelli di tutte le classi discendenti possono essere memorizzati. Inoltre siamo in grado di chiamare la stessa funzione per ogni elemento della matrice.

Esempio:

```
//--- Base class
class CShape
  {
protected:
   int            m_type;                 // Tipo di forma
   int            m_xpos;                 // X - coordinata del punto base
   int            m_ypos;                 // Y - coordinata del punto base
public:
   void           CShape(){m_type=0;};    // costruttore, type=0
   int            GetType(){return(m_type);};// restituisce il tipo della forma
virtual
   double         GetArea(){return (0); }// restituisce l'area della forma
  };
```

Ora, tutte le classi derivate hanno un membro di funzione getArea(), che restituisce un valore pari a zero. L'implementazione di questa funzione in ogni discendente varierà.

```
//--- La classe derivata Circle
class CCircle : public CShape            // Dopo i due punti si definisce la classe ba
  {                                      // da cui viene fatta l'eredità
private:
   double         m_radius;               // raggio del cerchio

public:
   void           CCircle(){m_type=1;};  // costruttore, type=1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// area del cerchio
  };
```

Per la classe Square la dichiarazione è la stessa:

```
//--- La classe derivata Square
```

```
class CSquare : public CShape             // Dopo i due punti si definisce la classe ba
  {                                       // da cui viene fatta l'eredità
private:
   double          m_square_side;         // lato del quadrato

public:
   void            CSquare(){m_type=2;}; // costruttore, type=1
   void            SetSide(double s){m_square_side=s;};
   virtual double  GetArea(){return (m_square_side*m_square_side);}// area del quadrat
  };
```

Per calcolare l'area del quadrato e del cerchio, abbiamo bisogno dei valori corrispondenti di m_radius e m_square_side, così abbiamo aggiunto la funzione setRadius e SetSide() nella dichiarazione della classe corrispondente.

Si presume che oggetti di tipo diverso (CCircle e CSquare) derivati da uno tipo base CShape sono utilizzati nel nostro programma. Il polimorfismo consente di creare un array di oggetti della classe base CShape, ma quando si dichiara questo array, questi oggetti sono ancora sconosciuti ed il loro tipo non è definito.

La decisione su quale tipo di oggetto sarà contenuto in ogni elemento della matrice sarà presa direttamente durante l'esecuzione del programma. Ciò comporta la creazione dinamica di oggetti di classi corrispondenti, e quindi la necessità di utilizzare puntatori agli oggetti invece degli oggetti.

Il nuovo operatore viene utilizzato per la creazione dinamica di oggetti. Ognuno di questi oggetti deve essere individualmente ed esplicitamente eliminato utilizzando l'operatore delete. Quindi noi dichiariamo un array di puntatori di tipo CShape, e creiamo un oggetto di un tipo adatto per ogni elemento (**nome_nuovaClasse**), come illustrato nello script di esempio seguente:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Dichiara un array di puntatori a oggetti del tipo base
   CShape *shapes[5];   // Un array di puntatori all'oggetto CShape

//--- Qui si riempie l'array con gli oggetti derivati
//--- Dichiarazione di un puntatore all'oggetto di tipo CCircle
   CCircle *circle=new CCircle();
//--- Impostazione proprietà dell'oggetto con il puntatore del cerchio
   circle.SetRadius(2.5);
//--- Piazza il valore del puntatore in shapes[0]
   shapes[0]=circle;

//--- Crea un altro oggetto CCircle e scrive il suo puntatore in shapes[1]
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);

//--- Qui "dimentichiamo" intenzionalmente di impostare un valore per shapes[2]
```

```
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- Imposta NULL per l'elemento che non è utilizzato
   shapes[2]=NULL;

//--- Crea un oggetto CSquare e scrivere il suo puntatore in shapes[3]
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- Crea un oggetto CSquare e scrivere il suo puntatore in shapes[4]
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- Abbiamo un array di puntatori, otteniamo la sua grandezza
   int total=ArraySize(shapes);
//--- Passaggio in un ciclo attraverso tutti i puntatori dell'array
   for(int i=0; i<5;i++)
     {
      //--- Se il puntatore in corrispondenza dell'indice specificato è valido
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- Log the type and square of the shape
         PrintFormat("The object of type %d has the square %G",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- Se il puntatore ha il tipo POINTER_INVALID
      else
        {
         //--- Notifica un errore
         PrintFormat("Le forme[%d] dell'oggetto non sono state inizializzate! Il suo p
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- Dobbiamo eliminare tutti gli oggetti dinamici creati
   for(int i=0;i<total;i++)
     {
//--- Possiamo eliminare solo gli oggetti con puntatori di tipo POINTER_DYNAMIC
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- Notifica di eliminazione
         PrintFormat("Elimina forme[%d]",i);
         //--- Elimina un oggetto dal suo puntatore
         delete shapes[i];
```

```
            }
        }
    }
```

Si prega di notare che quando si elimina un oggetto utilizzando l'operatore delete, il tipo del suo puntatore deve essere controllato. Solo gli oggetti con il puntatore POINTER_DYNAMIC possono essere eliminati usando delete. Per i puntatori di altro tipo, verrà restituito un errore.

Ma oltre la ridefinizione delle funzioni durante la ereditarietà, il polimorfismo include anche l' implementazione di una e le stesse funzioni con diversi set di parametri all'interno di una classe. Ciò significa che la classe può avere funzioni diverse con lo stesso nome, ma con un diverso tipo e/o insieme di parametri. In questo caso, il polimorfismo è attuato mediante l'overload di funzione.

Vedi anche

Standard Library

L' Overload

All'interno di una classe è possibile definire due o più metodi che utilizzano lo stesso nome, ma hanno un diverso numero di parametri. Quando ciò si verifica, i metodi vengono chiamati overloaded(_* sovraccarico) e tale processo viene indicato come metodo di overloading.

Il metodo di overloading è uno dei modi di realizzazione del polimorfismo. L' Overloading dei metodi è eseguito secondo le stesse regole dell' overloading di funzioni.

Se la funzione chiamata non ha alcuna corrispondenza esatta, il compilatore si mette in cerca di un'opportuna funzione su tre livelli sequenzialmente:

1. ricerca all'interno di metodi di classe;
2. ricerca all'interno dei metodi della classe di base, coerentemente dal più vicino antenato al primo.
3. ricerca tra le altre funzioni.

Se non vi è corrispondenza esatta a tutti i livelli, ma vengono trovate diverse funzioni adatte a diversi livelli, viene utilizzata la funzione trovata al livello minimo. All'interno di un livello, non vi può essere più di una funzione adatta.

Vedi anche

Funzioni di Ricarica

Funzioni Virtuali

La parola chiave virtuale è l'identificatore di funzione, che fornisce un meccanismo per selezionare dinamicamente in fase di esecuzione un appropriato membro-di-funzione tra le funzioni delle classi di base e derivate. Le strutture non possono avere funzioni virtuali. Esse possono essere utilizzate per cambiare le dichiarazioni solo per i membri della funzione.

La funzione virtuale, come una funzione ordinaria, deve avere un corpo eseguibile. Quando viene chiamata, la sua semantica è la stessa di quella delle altre funzioni.

Una funzione virtuale può essere sottoposta a override in una classe derivata. La scelta di quale definizione di funzione dovrebbe essere chiamata per una funzione virtuale, è fatta in modo dinamico (in fase di esecuzione). Un caso tipico è quando una classe base contiene una funzione virtuale, e classi derivate hanno le loro versioni di questa funzione.

Il puntatore alla classe base può indicare sia un oggetto di classe base che l'oggetto di una classe derivata. La scelta del membro-funzione da chiamare verrà eseguita in fase di esecuzione e dipende dal tipo di oggetto, non dal tipo di puntatore. Se non vi sono membri di un tipo derivato, la funzione virtuale della classe base viene utilizzata per impostazione predefinita.

I Distruttori sono sempre virtuali, indipendentemente dal fatto che siano dichiarati o meno con la parola chiave virtual.

Prendiamo in considerazione l'uso di funzioni virtuali sull'esempio di MT5_Tetris.mq5. La classe base CTetrisShape con la funzione virtuale Dra è definita nel file include MT5_TetrisShape.mqh.

```
//+------------------------------------------------------------------+
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
   void              Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) r
```

```
   virtual void      Draw()                     { return;               }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Inoltre, per ciascuna classe derivata, questa funzione è implementata in conformità con le caratteristiche di una classe discendente. Ad esempio, la prima forma CTetrisShape1 ha la propria implementazione della funzione Draw():

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- disegno forma
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- orizzontale
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
           }
        }
      else
        {
         //--- verticale
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
           }
        }
     }
  }
```

La forma Square è descritta dalla classe CTetrisShape6 ha la propria implementazione del metodo Draw():

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- Disegno forma
   virtual void      Draw()
```

```
     {
      int    i;
      string name;
      //---
      for(i=0; i<2; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
        }
      for(i=2; i<4; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
        }
     }
  };
```

A seconda della classe a cui appartiene l'oggetto creato, essa chiama la funzione virtuale di questa o quella classe derivata.

```
void CTetrisField::NewShape()
  {
//--- creazione di una delle 7 forme possibili a caso
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- draw
   m_shape.Draw();
//---
  }
```

Modifier 'override'

Il modificatore 'overrire' significa che la funzione dichiarata deve eseguire l'override("scavalco) del metodo di una classe genitore. L'utilizzo di questo metodo consente di evitare errori di primaria importanza, ad esempio, consente di evitare la modifica accidentale della firma del metodo. Supponiamo che, il metodo 'func' è definito nella classe base. Il metodo accetta una variabile int come argomento:

```
class CFoo
  {
   void virtual func(int x) const { }
  };
```

In seguito, il metodo viene scavalcato nella classe figlia:

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

Tuttavia, il tipo di argomento è erroneamente cambiato da int a short. In realtà, questo non è l'overriding del metodo, ma è l'overloading del metodo. Agendo in accordo all algoritmo definente la funzione overloadata, Il compilatore può in alcune situazioni scegliere un metodo definito nella classe base invece del metodo overridato.

Al fine di evitare tali errori, si dovrebbe aggiungere in modo esplicito il modificatore 'override' per il metodo che si desidera overridare.

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

Se la firma del metodo viene modificata durante l'override, il compilatore non sarà in grado di trovare un metodo con la stessa firma nella classe padre, e restituituirà un errore di compilazione:

```
'CBar::func' medoto è dichiarato con lo specificatore 'override' ma non fa l'override
```

Modificatore 'final'

Il modificatore 'final' fa il contrario — vieta il metodo prevalente alla classi figlie. Se un'implementazione metodo è sufficiente e completa del tutto, dichiarare questo metodo con il modificatore 'final' in modo da assicurarsi che non sarà modificato successivamente.

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

Se si tenta di eseguire l'override un metodo con il modificatore 'final', come mostrato nell'esempio precedente, il compilatore restituirà un errore:

```
'CFoo::func' metodo dichiarato come 'final' non può essere overridato da 'CBar::func'
vedere la dichiarazione di 'CFoo::func'
```

Vedi anche

Standard Library

I Membri Statici di una Classe/Struttura

I Membri Statici

The members of a class can be declared using the storage class modifier static. Questi dati membri sono condivisi da tutte le istanze di quella classe e sono memorizzati in un unico posto. Dati membri non-statici vengono creati per ciascuna variabile dell' oggetto della classe.

L'incapacità di dichiarare membri statici di una classe avrebbe comportato la necessità di dichiarare questi dati a livello globale del programma. Romperebbe la relazione tra i dati e la loro classe, e non è coerente con il paradigma di base della programmazione orientata agli oggetti - unione dati e metodi per la loro gestione in una classe. Il membro statico consente ai dati della classe che non sono specifici di una particolare istanza di esistere nell'ambito classe.

Poiché un membro della classe statica non dipende dalla particolare istanza, il riferimento è il seguente:

```
class_name::variable
```

dove *class_name* è il nome della classe, e *variabile* è il nome del membro della classe.

Come vedete, per accedere al membro statico di una classe, viene utilizzato l'operatore di risoluzione del contesto :. Quando si accede ad un membro statico all'interno dei metodi della classe, l'operatore di contesto è opzionale.

Il membro statico di una classe deve essere esplicitamente inizializzato con il valore desiderato. Per questo esso deve essere dichiarato ed inizializzato in ambito globale. La sequenza di inizializzazione dei membri statici corrisponderà alla sequenza della loro dichiarazione in ambito globale.

Ad esempio, abbiamo la classe *CParser* usata per l'analisi del testo, ed abbiamo bisogno di contare il numero totale di parole e caratteri elaborati. Abbiamo solo bisogno di dichiarare i necessari membri della classe di statici ed inizializzarli a livello globale. Poi tutte le istanze della classe utilizzeranno i contatori comuni di parole e caratteri.

```
//+------------------------------------------------------------------------------+
//| Classe "Text analyzer"                                                       |
//+------------------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
 //--- Costruttore e distruttore
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- Inizializzazione dei membri statici della classe Parser a livello globale
int CParser::s_words=0;
int CParser::s_symbols=0;
```

Un membro statico della classe può essere dichiarato con la parola chiave*const*. Tali costanti statiche devono essere inizializzate a livello globale con la parola chiave *const* :

```
//+------------------------------------------------------------------------------+
//| Classe "Stack" per memorizzare i dati elaborati                              |
//+------------------------------------------------------------------------------+
class CStack
  {
public:
                     CStack(void);
                    ~CStack(void){};
...
private:
   static const int  s_max_length; // Massima capacità stack
  };

//--- Inizializzazione delle costanti statiche della classe CStack
const int CStack::s_max_length=1000;
```

Puntatore this

La parola chiave this denota un implicitamente dichiarato puntatore a sé - ad un'istanza specifica della classe, nel contesto del quale viene eseguito il metodo. Esso può essere utilizzato solo in metodi non statici della classe. Il puntatore this è un implicito non-statico membro di qualsiasi classe.

Nelle funzioni statiche è possibile accedere solo a membri/metodi statici di una classe.

Metodi statici

In MQL5 possono essere utilizzate le funzioni membro del tipo static. Il modificatore *static* deve precedere il tipo di ritorno di una funzione nella dichiarazione all'interno di una classe.

```
class CStack
  {
public:
 //--- Costruttore e distruttore
                     CStack(void){};
                    ~CStack(void){};
 //--- Massima capacità stack
   static int        Capacity();
private:
   int               m_length;     // Il numero di elementi nello stack
   static const int  s_max_length; // Massima capacità stack
  };
//+------------------------------------------------------------------------------+
//| Restituisce il numero massimo di elementi da memorizzare nello stack         |
//+------------------------------------------------------------------------------+
int CStack::Capacity(void)
  {
   return(s_max_length);
```

```
  }
//--- Inizializzazione delle costanti statiche della classe CStack
const int CStack::s_max_length=1000;
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- dichiara il tipo di variabile CStack
   CStack stack;
//--- chiama il metodo statico dell'oggetto
   Print("CStack.s_max_length=",stack.Capacity());
//--- può anche essere chiamato nel seguente modo, giacchè il metodo è statico e non
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

Un metodo con il modificatore const viene chiamato costante e non può modificare i membri impliciti della sua classe. Le dichiarazione di funzioni costanti di una classe e parametri costanti, viene denominata controllo di *correttezza-const*. Attraverso questo controllo si può essere sicuri che il compilatore garantisce la coerenza dei valori degli oggetti e restituisce un errore in fase di compilazione se vi è qualcosa di sbagliato.

Il modificatore const viene inserito dopo l'elenco degli argomenti all'interno di una dichiarazione della classe. La definizione fuori da una classe deve anche includere il modificatore *const*:

```
//+------------------------------------------------------------------------------------+
//| Classe "Rettangolo"                                                                |
//+------------------------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;      // Spessore
   double            m_height;     // Altezza
public:
   //--- Costruttore e distruttore
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- Calcola l'area
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------------------------+
//| Restituisce l'area dell'oggetto "Rettangolo"                                       |
//+------------------------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
//+------------------------------------------------------------------------------------+
```

```
//| Restituisce il prodotto di due variabili                         |
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- crea un rettangolo con i lati uguali a 5 e a 6
   CRectangle rect(5,6);
//--- trova l'area del rettangolo utilizzando un metodo costante
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- trova il prodotto deei numeri con il metodo statico della classe CRectangle
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

Un ulteriore argomento a favore dell'utilizzo del controllo di costanza è il fatto che in questo caso, il compilatore genera un'ottimizzazione speciale, per esempio, inserisce un oggetto costante nella memoria di sola lettura.

Una funzione statica non può essere determinata con il modificatore const, perché questo modificatore assicura la costanza dei membri dell' istanza quando si chiama questa funzione. Ma, come detto sopra, la funzione statica non può accedere a membri non statici della classe.

Vedi anche

Variabili statiche, Variabili, Riferimenti. Modificatore & e Parola Chiave this

Templates di Funzioni

Funzioni overloadate vengono comunemente utilizzati per eseguire operazioni simili su vari tipi di dati. ArraySize() è un semplice esempio di tale funzione in MQL5. Restituisce la grandezza di qualsiasi tipo di array. In realtà, questa funzione di sistema è overloadata e l'intera implementazione di un tale sovraccarico è nascosto agli sviluppatori di applicazioni MQL5:

```
int  ArraySize(
   void&  array[]      // array verificato
   );
```

Ciò significa che il compilatore del linguaggio MQL5 inserisce implementazioni necessarie per ogni chiamata di questa funzione. Ad esempio, è così che si può fare per gli array di tipo double:

```
int  ArraySize(
   int&  array[]      // array con elementi di tipo int
   );
```

La funzione Arraysize() può essere visualizzata nel modo seguente per il tipo di array MqlRates per l'utilizzo di quotazioni in formato dati storici:

```
int  ArraySize(
   MqlRates&  array[] // array riempito con valori di tipo MqlRates
   );
```

Pertanto, è molto conveniente utilizzare la stessa funzione per lavorare con differenti tipi. Ad ogni modo, dev' essere effettuato tutto il lavoro preliminare - la funzione necessaria deve essere overloadata per tutti i tipi di dati con cui dovrebbe funzionare correttamente.

C'è una soluzione conveniente. Se operazioni simili devono essere eseguite per ogni tipo di dati, è possibile utilizzare i modelli di funzione. In questo caso, un programmatore ha bisogno di scrivere solo una descrizione del template funzione. Quando si descrive il template in modo tale, dovremmo specificare solo qualche parametro formale invece di un tipo definito di dati con cui la funzione dovrebbe lavorare. Il compilatore genera automaticamente varie funzioni per la corretta gestione di ogni tipo in base ai tipi di argomenti utilizzati quando si chiama la funzione.

La definizione del template di funzione inizia con la parola chiave template seguita dalla lista dei parametri formali tra parentesi angolari. Ogni parametro formale è preceduto dalla parola chiave typename. I tipi di parametri formali sono di tipo built-in o definiti-dall'utente. Essi sono utilizzati:

- per specificare i tipi di argomenti della funzione,
- per specificare il tipo di valore di ritorno della funzione,
- per dichiarare le variabili all'interno della definizione di funzione

Il numero di parametri del modello non può essere superiore ad otto. Ogni parametro formale nella definizione del modello dovrebbe apparire nella lista dei parametri della funzione almeno una volta. Ogni nome del parametro formale deve essere univoco.

Di seguito è riportato un esempio di un modello di funzione per ricercare il valore più alto nell' array di qualsiasi tipo numerico (numeri interi e reali):

```
template<typename T>
T ArrayMax(T &arr[])
  {
   uint size=ArraySize(arr);
   if(size==0) return(0);

   T max=arr[0];
   for(uint n=1;n<size;n++)
      if(max<arr[n]) max=arr[n];
//---
   return(max);
  }
```

Questo template definisce la funzione che rileva il valore massimo nell'array passato e restituisce questo valore come risultato. Tenete a mente che la funzione ArrayMaximum() costruita in MQL5 restituisce solo l'indice di più alto valore che può essere utilizzato per trovare il valore stesso. Ad esempio:

```
//--- crea un array
   double array[];
   int size=50;
   ArrayResize(array,size);
//---  riempie con valori casuali
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//--- trova la posizione del valore più elevato nell'array
   int max_position=ArrayMaximum(array);
//--- ora ottiene il valore stesso più elevato, nell'array
   double max=array[max_position];
//--- mostra il valore trovato
   Print("Max value = ",max);
```

Così, abbiamo effettuato due passaggi per ottenere il più alto valore nell'array. Con il template di funzione ArrayMax(), si può ottenere il risultato del tipo necessario semplicemente passando l'array di un tipo appropriato in questa funzione. Significa che invece delle ultime due righe

```
//--- trova la posizione del valore più elevato nell'array
   int max_position=ArrayMaximum(array);
//--- ora riceve il valore stesso più elevato, nell'array
   double max=array[max_position];
```

ora è possibile utilizzare una sola riga, in cui il risultato restituito ha lo stesso tipo dell'array passato nella funzione:

```
//--- trova il valore più alto
   double max=ArrayMax(array);
```

In questo caso, il tipo di risultato restituito dalla funzione ArrayMax() funzione abbina automaticamente il tipo di matrice.

Utilizzare la parola chiave typename per ottenere il tipo di argomento come stringa al fine di creare i metodi di uso generale di lavoro con diversi tipi di dati. Consideriamo un esempio specifico della funzione che restituisce tipo di dati come una stringa:

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------------------+
//|                                                                              |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: type=",GetTypeName(d_value), ",   value=", d_value);
   Print("i_value: type=",GetTypeName(i_value), ",   value=", i_value);
   Print("trade: type=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------------------+
//| Il tipo viene restituito come una linea                                      |
//+------------------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- restituisce il tipo come una linea
   return(typename(T));
//---
  }
```

Templates di funzioni possono anche essere utilizzati per metodi della classe, per esempio:

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

Templates di funzioni non devono essere dichiarati con le parole chiave export, virtual e #import.

Overload della funzione template

Un sovraccarico di funzione template può essere necessario a volte. Per esempio, abbiamo una funzione template che scrive il valore del secondo parametro nel primo utilizzando typecasting. MQL5 non consente typecasting string a bool. Possiamo farlo noi stessi - creiamo un sovraccarico di una funzione template. Per esempio:

```
//+------------------------------------------------------------------+
//| Funzione Template                                                |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Speciale overload per bool+string                                |
//+------------------------------------------------------------------+
string Assign(bool &var1,string var2)
  {
   var1=(StringCompare(var2,"true",false) || StringToInteger(var2)!=0);
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
   int i;
   bool b;
   Print(Assign(i,"test"));
   Print(Assign(b,"test"));
  }
```

Come risultato dell'esecuzione del codice, possiamo vedere che la funzione template Assign() è stata utilizzata per la coppia int+string, mentre la versione overload è già stata utilizzata per la coppia string+bool durante la seconda chiamata.

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

Guarda anche

Overload

Vantaggi dei templates

I Templates delle funzioni vengono utilizzati quando è necessario eseguire operazioni simili su vari tipi di dati, ad esempio, la ricerca di un elemento massimo nell'array. Il vantaggio principale di applicare i templates è che non c'è bisogno di codare un separato overload per ogni tipo. Invece di dichiarare multiple overloads di ogni tipo

```
double ArrayMax(double array[])
  {
   ...
  }
int ArrayMax(int array[])
  {
   ...
  }
uint ArrayMax(uint array[])
  {
   ...
  }
long ArrayMax(long array[])
  {
   ...
  }
datetime ArrayMax(datetime array[])
  {
   ...
  }
```

abbiamo bisogno di scrivere una sola funzione template

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

per utilizzarlo nel codice:

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

Qui, il parametro formale *T* specificante un tipo di dati utilizzato è sostituito con un tipo effettivamente applicata durante la compilazione, cioè il compilatore genera automaticamente una funzione separata per ogni tipo - double, datetime, eccetera. MQL5 permette inoltre di sviluppare modelli di classi utilizzando tutti i vantaggi di questo approccio.

Templates delle Classi

Un template(modello) della classe è dichiarato con la parola chiave template seguita da parentesi angolari<> enumeranti la lista dei parametri formali con la parola chiave typename. Questa voce informa il compilatore che si tratta di una classe generica con il parametro formale *T* che definisce un vero e proprio tipo di variabile in sede di attuazione di una classe. Ad esempio, creiamo una classe vettore per la memorizzazione di un array con elementi di tipo *T*:

```
#define TOSTR(x) #x+" "   // macro per visualizzare il nome dell'oggetto
//+------------------------------------------------------------------+
//| Classe vettore per memorizzare elementi tipo-T                   |
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- il costruttore crea un'array per 10 elementi di default
   void TArray(void){ArrayResize(m_array,10);}
   //--- il costruttore per creare un vettore con una grandezza array specificata
   void TArray(int size){ArrayResize(m_array,size);}
   //--- restituisce il tipo ed ammontare di dati conservati nell'oggetto di tipo TAr
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

Quindi, cerchiamo di applicare metodi diversi per creare tre oggetti *TArray* nel programma per lavorare con vari tipi

```
void OnStart()
  {
   TArray<double> double_array;   // il vettore ha la grandezza di default di 10
   TArray<int> int_array(15);     // il vettore ha la grandezza di 15
   TArray<string> *string_array;  // puntatore al vettore TArray<string>
//--- crea un oggetto dinamico
   string_array=new TArray<string>(20);
//--- mostra il nome dell'oggetto, il tipo di dati e grandezza vettore nel Journal
   PrintFormat("%s (%s)",TOSTR(double_array),double_array.Type());
   PrintFormat("%s (%s)",TOSTR(int_array),int_array.Type());
   PrintFormat("%s (%s)",TOSTR(string_array),string_array.Type());
//--- rimuove un oggetto dinamico prima di completare il programma
   delete(string_array);
  }
```

Risultati esecuzione script:

```
  double_array  (double:10)
  int_array  (int:15)
  string_array  (string:20)
```

Ora, abbiamo 3 vettori con diversi tipi di dati: double, int e string.

I modelli(template) della classe sono adatti per lo sviluppo di contenitori - oggetti progettati per incapsulare altri oggetti di qualsiasi tipo. Gli oggetti container sono raccolte già contenenti oggetti di un certo tipo. Di solito, il lavoro con i dati memorizzati è immediatamente costruito nel contenitore.

Ad esempio, è possibile creare un modello della classe che non consente l'accesso a un elemento al di fuori dell'array, evitando così l' errore critico "out of range" .

```
//+------------------------------------------------------------------+
//| Classe per il libero accetto agli elementi dell'array            |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- costruttore default
   void              TSafeArray(void){}
   //--- costruttore per creare l'array di grandezza specificata
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- grandezza array
   int               Size(void){return(ArraySize(m_array));}
   //--- cambia la grandezza array
   int               Resize(int size,int reserve){return(ArrayResize(m_array,size,rese
   //--- rilascia l'array
   void              Erase(void){ZeroMemory(m_array);}
   //--- operatore per accedere agli elementi dell'array per indice
   T                 operator[](int index);
   //--- assegnazione dell'operatore per ricevere tutti gli elementi dall'array tutto
   void              operator=(const T  &array[]); // T type array
  };
//+------------------------------------------------------------------+
//| Ricezione di un elemento per indice                              |
//+------------------------------------------------------------------+
template<typename T>
T TSafeArray::operator[](int index)
  {
   static T invalid_value;
//---
   int max=ArraySize(m_array)-1;
   if(index<0 || index>=ArraySize(m_array))
     {
      PrintFormat("%s indice %d non è nel range (0-%d)!",__FUNCTION__,index,max);
      return(invalid_value);
     }
//---
   return(m_array[index]);
```

```
  }
//+------------------------------------------------------------------+
//| Assegnazione per l'rray                                          |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- Tipo T deve supportare la copia dell'operatore
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- copia l'array delle quotazioni
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) ha restituito il codice errore %d",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- crea un contenintore ed inserisce i valori MqlRates in esso
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- indice dentro l'array
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- indice fuori l'array
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

Si prega di notare che la dichiarazione del template dev essere utilizzata anche quando si descrivono i metodi fuori della dichiarazione della classe:

```
template<typename T>
T TSafeArray::operator[](int index)
  {
   ...
  }
template<typename T>
void TSafeArray::operator=(const T  &array[])
```

```
  {
   ...
  }
```

Modelli della classe e funzionalità consentono di definire più parametri formali separati da virgole, ad esempio, la raccolta Map per la memorizzazione di coppie "valori-chiave":

```
template<typename Key, template Value>
class TMap
  {
   ...
  }
```

Guarda anche

Templates funzioni, Overload

Classi Astratte e Funzioni Virtuali Pure

Le classi astratte vengono utilizzate per la creazione di entità generiche, che prevedi di utilizzare per creare classi derivate più specifiche. Una classe astratta può essere utilizzata solo come classe base per un'altra classe, motivo per cui è impossibile creare un oggetto del tipo di classe astratta.

Una classe che contiene almeno una pura funzione virtuale è astratta. Pertanto, le classi derivate dalla classe astratta devono implementare tutte le sue pure funzioni virtuali, altrimenti saranno anche classi astratte.

Una funzione virtuale viene dichiarata "pura" utilizzando la sintassi degli specificatori puri. Si consideri l'esempio della classe CAnimal, che viene creata solo per fornire funzioni comuni - gli oggetti del tipo CAnimal sono troppo generici per l'uso pratico. Quindi, CAnimal è un buon esempio per una classe astratta:

```
class CAnimal
  {
public:
                     CAnimal();      // Constructor
   virtual void      Sound() = 0;    // Una funzione virtuale pura
private:
   double            m_legs_count;   // Il numero di gambe dell'animale
  };
```

Qui Sound() è una pura funzione virtuale, perché è dichiarata con l'identificatore della funzione virtuale pura PURE (=0).

Le funzioni virtuali pure sono solo le funzioni virtuali per le quali è impostato lo specificatore PURE: (=NULL) o (=0). Esempio di dichiarazione di classe astratta e uso:

```
class CAnimal
  {
public:
   virtual void      Sound()=NULL;   // IL metodo PURE, dovrebbe essere sovrascritto
  };
//--- Derivato da una classe astratta
class CCat : public CAnimal
 {
public:
  virtual void       Sound() { Print("Myau"); } // PURE è sovrascritto, CCat non è as
 };

//--- Esempi di uso errato
new CAnimal;          // Errore di 'CAnimal' - il compilatore restituisce l'errore "imp
CAnimal some_animal; // Errore di 'CAnimal': il compilatore restituisce l'errore "impo

//--- Esempi di uso corretto
new CCat;// Nessun errore: la classe CCat non è astratta
CCat cat;// Nessun errore: la classe CCat non è astratta
```

Restrizioni su classi astratte

Se il costruttore di una classe astratta chiama una funzione virtuale pura (direttamente o indirettamente), il risultato non è definito.

```
//+------------------------------------------------------------------+
//| Una classe base astratta                                         |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- Una funzione virtuale pura
   virtual void      Sound(void)=NULL;
   //--- Funzione
   void              CallSound(void) { Sound(); }
   //--- Costruttore
   CAnimal()
    {
     //--- Una chiamata esplicita del metodo virtuale
     Sound();
     //--- Una chiamata implicita (utilizzando una terza funzione)
     CallSound();
     //--- Un costruttore e/o un distruttore chiama sempre le sue funzioni,
     //--- anche se sono virtuali e sovrascritti da una funzione chiamata in una class
     //--- Se la funzione chiamata è virtuale pura
     //--- la sua chiamata causerà un errore critico di runtime: "pura chiamata di fu
    }
  };
```

Tuttavia, costruttori e distruttori per le classi astratte possono chiamare altri membri di funzione.

Namespaces

Un namespace è un'area dichiarata appositamente, all'interno della quale sono definiti vari ID: variabili, funzioni, classi, ecc. Si imposta usando la parola chiave namespace parola chiave:

```
namespace nome dello_spazio {
// elenco di definizioni di funzioni, classi e variabili
}
```

L'applicazione del 'namespace' consente di suddividere il namespace globale in subspaces(sottospazi). Tutti gli ID all'interno del namespace sono disponibili gli uni agli altri senza una specifica. L' operatore :: (operazione di risoluzione del contesto) viene utilizzato per accedere ai membri del namespace dall'esterno.

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
}
//+------------------------------------------------------------------+
//| Funzione di avvio del programma di script |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- lavorare con il namespace ProjectData
ProjectData::DataManagermgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

I namespaces vengono utilizzati per disporre un codice sotto forma di gruppi logici ed evitare conflitti di nomi che possono verificarsi quando si utilizzano più librerie in un programma. In tali casi, ciascuna libreria può essere dichiarata nel suo namespace per accedere esplicitamente alle funzioni ed alle classi necessarie di ciascuna libreria.

Un namespace può essere dichiarato in più blocchi in uno o più file. Il compilatore combina tutte le parti insieme durante il preprocessing(preelaborazione) ed il namespace risultante contiene tutti i membri dichiarati in tutte le parti. Supponiamo di avere una classe implementata nel file include Sample.mqh:

```
//+------------------------------------------------------------------+
//|                                                   Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
                     A() {Print(__FUNCTION__);}
```

```
  };
```

Vogliamo usare questa classe nel nostro progetto, ma abbiamo già una classe A. Per poter utilizzare entrambe le classi ed evitare conflitti di ID, è sufficiente avvolgere il file incluso in un namespace:

```
//--- dichiara la prima classe A.
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
//--- avvolge la classe A dal file Sample.mqh nel namespace Library per evitare un cor
namespace Library
{
#include "Sample.mqh"
}
//--- aggiunge ancora un'altra classe al namespace Library
namespace Library
{
class B
  {
public:
                     B() {Print(__FUNCTION__);}
  };
}
//+------------------------------------------------------------------+
//| Funzione di avvio del programma di script |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- usa la classe A dal namespace globale
   A a1;
//--- usa le classi A e B dal namespace Library
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
Risultato:
   A::A
   Library::A::A
   Library::B::B
*/
```

I namespaces possono essere nidificati. Uno namespace nidificato ha accesso illimitato ai membri del suo spazio genitore, ma i membri dello spazio genitore non hanno accesso illimitato al namespace nidificato.

```
namespace General
```

```
{
int Func();

namespace Details
{
 int Counter;
 int Refresh()  {return Func(); }
}

int GetBars()   {return(iBars(Symbol(), Period()));};
int Size(int i) {return Details::Counter;}
}
```

Namespace Globale

Se l'ID non viene dichiarato esplicitamente nel namespace, viene considerato una parte implicita del namespace globale. Per impostare l'ID globale in modo esplicito, utilizzare l'operatore di risoluzione dell'ambito senza un nome. Ciò ti consentirà di distinguere questo ID da qualsiasi altro elemento con lo stesso nome situato in un namespace diverso. Ad esempio, quando si importa una funzione:

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//| Alcune funzioni |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma di script |
//+------------------------------------------------------------------+
void OnStart()
  {
//+--- chiama la funzione importata
   Print(lib::Func());
//+--- chiama la nostra funzione
   Print(::Func());
  }
```

In questo caso, tutte le funzioni importate dalla funzione DLL sono state incluse nel namespace con lo stesso nome. Ciò ha permesso al compilatore di determinare chiaramente la funzione da chiamare.

Guarda anche

Variabili globali, Variabili locali, Ambito di Visibilità e Durata delle Variabili, Creazione ed Eliminazione di Oggetti

Costanti, Enumerazioni e Strutture

Per semplificare la scrittura di programmi e rendere i testi del programma più convenienti per la percezione, il linguaggio MQL5 fornisce costanti standard predefinite ed enumerazioni. Oltre a questo, strutture di servizio vengono utilizzate per la memorizzazione di informazioni.

Costanti standard sono simili a macros e sono di tipo int.

Le costanti sono raggruppate per i loro scopi:

- Costanti del grafico vengono utilizzate quando si lavora con prezzi dei grafici: apertura, navigazione, impostazione dei parametri;
- Costanti oggetti sono destinati alla trasformazione di oggetti grafici che possono essere creati e visualizzati nei grafici;
- Costanti Indicatore vengono utilizzate per lavorare con indicatori standard e personalizzati;
- Stato dell'ambiente costanti descrivono le proprietà di un programma-MQL5, mostrano informazioni su un terminale client, strumento finanziario ed il corrente account;
- Costanti trade consentono di specificare una varietà di informazioni in corso di trading;
- Costanti nominate sono costanti del linguaggio MQL5;
- Strutture di dati descrivono i formati di memorizzazione dei dati utilizzati;
- I codici di errori e avvisi descrivono i messaggi del compilatore e le risposte del trading server alle richieste di trade,
- Costanti In/out sono progettate per lavorare con file di funzioni e la visualizzazione di messaggi sullo schermo dalla funzione MessageBox().

Costanti del Grafico

Costanti che descrivono varie proprietà di grafici sono suddivise nei seguenti gruppi:

- Tipi di eventi - eventi che si verificano quando si lavora con i grafici;
- Timeframes dei grafici - periodi built-in standard;
- Proprietà del grafico - identificatori utilizzati come parametri di funzioni grafico;
- Costanti Posizionamento - valore del parametro della funzione ChartNavigate();
- Visualizzazione dei grafici - imposta l'aspetto del grafico.

Tipi di Eventi del Grafico

Ci sono 11 tipi di eventi che possono essere elaborati utilizzando la funzione predefinita OnChartEvent(). Per gli eventi personalizzati vengono forniti 65535 identificatori nella gamma di CHARTEVENT_CUSTOM fino a CHARTEVENT_CUSTOM_LAST compreso. Per generare un evento personalizzato, deve essere utilizzata la funzione EventChartCustom().

ENUM_CHART_EVENT

ID	Descrizione
CHARTEVENT_KEYDOWN	Tasti
CHARTEVENT_MOUSE_MOVE	Movimento del mouse, clic del mouse (se CHART_EVENT_MOUSE_MOVE=true è impostato per il grafico)
CHARTEVENT_MOUSE_WHEEL	Pressione o scorrimento della rotellina del mouse (se CHART_EVENT_MOUSE_WHEEL=True per il chart)
CHARTEVENT_OBJECT_CREATE	Oggetto grafico creato (se CHART_EVENT_OBJECT_CREATE=true è impostato per il chart)
CHARTEVENT_OBJECT_CHANGE	Oggetto grafico proprietà modificata tramite la finestra delle proprietà
CHARTEVENT_OBJECT_DELETE	Oggetto grafico eliminato (se CHART_EVENT_OBJECT_DELETE=true è impostato per il chart)
CHARTEVENT_CLICK	Facendo clic su un chart
CHARTEVENT_OBJECT_CLICK	Facendo clic su un oggetto grafico
CHARTEVENT_OBJECT_DRAG	Drag and drop di un oggetto grafico
CHARTEVENT_OBJECT_ENDEDIT	Fine di modifica del testo nella modifica degli oggetti grafici
CHARTEVENT_CHART_CHANGE	Modifica della grandezza del grafico o modifica delle proprietà del grafico tramite la finestra delle Proprietà
CHARTEVENT_CUSTOM	Numero iniziale di un evento da una serie di eventi personalizzati
CHARTEVENT_CUSTOM_LAST	Il numero finale di un evento di una serie di eventi personalizzati

Per ciascun tipo di evento, i parametri di input della funzione OnChartEvent() hanno valori definiti che sono necessari per l'elaborazione di questo evento. Gli eventi ed i valori passati attraverso questi parametri sono elencati nella tabella seguente.

Evento	Valore del parametro id	Valore del parametro lparam	Valore del parametro dparam	Valore del parametro sparam
Evento di una sequenza di tasti	CHARTEVENT_KEYDOWN	codice di un tasto premuto	Ripete in conteggio (il numero di volte che il tasto viene ripetuto come risultato dell'utente che tiene premuto il tasto)	Il valore di stringa di una maschera di bit che descrive lo status dei pulsanti della tastiera
Eventi del mouse (se CHART_EVENT_MOUSE_MOVE=true è impostato per il grafico)	CHARTEVENT_MOUSE_MOVE	la coordinata X	la coordinata Y	Il valore di stringa di una maschera di bit che descrive lo stato dei pulsanti del mouse
Evento rotellina del mouse (se CHART_EVENT_MOUSE_WHEEL=true per il chart)	CHARTEVENT_MOUSE_WHEEL	Flag di status di tasti e pulsanti del mouse, le coordinate X e Y del puntatore del mouse. Vedere la descrizione nell' esempio sotto	Il valore Delta della rotellina del mouse	—
evento di creazione di oggetti grafici (se CHART_EVENT_OBJECT_CREATE=true è impostato per il grafico)	CHARTEVENT_OBJECT_CREATE	—	—	Nome dell'oggetto grafico creato
Evento di cambiamento di proprietà di un oggetto attraverso la finestra delle proprietà	CHARTEVENT_OBJECT_CHANGE	—	—	Nome dell'oggetto grafico modificato
Evento di eliminazione oggetto grafico (Se CHART_EVENT_OBJECT_DELETE=t	CHARTEVENT_OBJECT_DELETE	—	—	Nome dell'oggetto grafico eliminato

Evento	Valore del parametro id	Valore del parametro lparam	Valore del parametro dparam	Valore del parametro sparam
rue è impostato per il grafico)				
Evento di un click del mouse sul chart	CHARTEVENT_CLICK	la coordinata X	la coordinata Y	—
Evento di un clic del mouse in un oggetto grafico appartenente alla tabella	CHARTEVENT_OBJECT_CLICK	la coordinata X	la coordinata Y	Nome dell'oggetto grafico, in cui l'evento si è verificato
Evento di trascinamento di un oggetto grafico con il mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome dell'oggetto grafico spostato
Evento di fine modifica del testo nella casella di immissione dell'oggetto grafico LabelEdit	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome dell' oggetto grafico LabelEdit, in cui la modifica del testo è stata completata
Evento di cambiamento della grandezza del chart o modifica delle proprietà del chart tramite la finestra delle Proprietà	CHARTEVENT_CHART_CHANGE	—	—	—
ID dell'evento dell'utente con il numero N	CHARTEVENT_CUSTOM+N	Valore impostato dalla funzione EventChartCustom()	Valore impostato dalla funzione EventChartCustom()	Valore impostato dalla funzione EventChartCustom()

Esempio:

```
#define KEY_NUMPAD_5        12
#define KEY_LEFT            37
#define KEY_UP              38
#define KEY_RIGHT           39
#define KEY_DOWN            40
```

```
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                          |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("L'expert con nome ",MQL5InfoString(MQL5_PROGRAM_NAME)," sta girando");
//--- abilita gli eventi di creazione degli oggetti
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- abilita gli eventi di eliminazione degli oggetti
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- l'aggiornamento forzato delle proprietà del chart garantisce la preparazione per
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| ChartEvent function                                                                |
//+------------------------------------------------------------------------------------+
void OnChartEvent(const int id,         // Identificatore eventi
                  const long& lparam,   // Parametro evento di tipo long
                  const double& dparam, // Parametro evento di tipo double
                  const string& sparam  // Parametro evento di tipo stringa
                  )
  {
//--- il bottone sinistro del mouse è stato premuto sul chart
   if(id==CHARTEVENT_CLICK)
     {
      Print("Le coordinate del click del mouse sul chart sono: x = ",lparam,"  y = ",d
     }
//--- il mouse è stato cliccato sull'oggetto grafico
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("Il mouse è stato cliccato sull'oggetto con nome'"+sparam+"'");
     }
//--- il tasto è stato premuto
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("E' stato premuto KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("E' stato premuto KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("E' stato premuto KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("E' stato premuto il tasto KEY_UP");
```

```
         case KEY_NUMLOCK_RIGHT: Print("E' stato premuto il tasto KEY_NUMLOCK_RIGHT");
         case KEY_RIGHT:         Print("E' stato premuto il tasto KEY_RIGHT");
         case KEY_NUMLOCK_DOWN:  Print("E' stato premuto il tasto KEY_NUMLOCK_DOWN");
         case KEY_DOWN:          Print("E' stato premuto il tasto KEY_DOWN");
         case KEY_NUMPAD_5:      Print("E' stato premuto il tasto KEY_NUMPAD_5");
         case KEY_NUMLOCK_5:     Print("E' stato premuto il tasto KEY_NUMLOCK_5 ");
         default:                Print("Alcuni tasti non elencati sono stati premuti")
        }
      ChartRedraw();
     }
//--- l'oggetto è stato eliminato
   if(id==CHARTEVENT_OBJECT_DELETE)
     {
      Print("L'oggetto con nome ",sparam," è stato premuto");
     }
//--- L'oggetto è stato creato
   if(id==CHARTEVENT_OBJECT_CREATE)
     {
      Print("L'oggetto con nome ",sparam," è stato creato");
     }
//--- l'oggetto è stato mosso o le sue coordinate dei punti di ancoraggio sono state c
   if(id==CHARTEVENT_OBJECT_DRAG)
     {
      Print("Le coordinate dei punti di ancoraggio dell'oggetto con nome ",sparam," sc
     }
//--- il testo nell'Edit dell'oggetto è stato cambiato
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
     {
      Print("Il testo nel campo Edit dell'oggetto con nome ",sparam," è stato cambiato
     }
  }
```

Per l'evento CHARTEVENT_MOUSE_MOVE il parametro stringa **@param** contiene informazioni sullo stato della tastiera e dei pulsanti del mouse:

Bit	Descrizione
1	Stato del tasto sinistro del mouse
2	Stato del tasto destro del mouse
3	Stato del pulsante SHIFT
4	Stato del pulsante CTRL
5	Stato del pulsante centrale del mouse
6	Stato del primo pulsante del mouse in più
7	Stato del secondo tasto del mouse in più

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- abilita messaggi CHART_EVENT_MOUSE_MOVE
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
// --- disabilita il menu contestuale del grafico (a destra)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
// --- disabilita il mirino (tramite il pulsante centrale)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
//--- l'aggiornamento forzato delle proprietà del chart garantisce la preparazione pe
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse sinistra
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse destra
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse centro
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse primo tasto X
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse secondo tasto X
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // tasto shift
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // tasto control
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

Per l'evento CHARTEVENT_MOUSE_WHEEL, i parametri **lParam** e **dparam** contengono informazioni sugli stati dei tasti **Ctrl** e **Shift**, dei pulsanti del mouse, delle coordinate del cursore e del valore di scorrimento della rotellina del mouse. Per una migliore comprensione, esegui questo Expert Advisor su un chart e scorri la rotellina del mouse, mentre premi diversi bottoni e tieni premuti i tasti descritti nel codice.

Esempio diCHARTEVENT_MOUSE_WHEEL event processing**:**

```
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
```

```
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Abilitazione dei messaggi di scorrimento della rotellina del mouse
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- L'aggiornamento forzato delle proprietà del chart garantisce la preparazione per
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- Considera lo stato dei pulsanti e della rotella del mouse per questo event
      int flg_keys = (int)(lparam>>32);          // La flag degli stati dei tasti Ctrl
      int x_cursor = (int)(short)lparam;         // la coordinata X dove si è verifica
      int y_cursor = (int)(short)(lparam>>16);   // la coordinata Y dove si è verifica
      int delta    = (int)dparam;                // il valore totale dello scroll del
      //--- Elaborazione della flag
      string str_keys="";
      if((flg_keys&0x0001)!=0) str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0) str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0) str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0) str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0) str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0) str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0) str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1) + "'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
  }
//+------------------------------------------------------------------+ /*
   Esempio di output
   CHARTEVENT_MOUSE_WHEEL: Ctrl pressed: X=193, Y=445, delta=-120
   CHARTEVENT_MOUSE_WHEEL: Shift pressed: X=186, Y=446, delta=120
   CHARTEVENT_MOUSE_WHEEL:  X=178, Y=447, delta=-120
   CHARTEVENT_MOUSE_WHEEL:  X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: MiddleButton pressed: X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: LeftButton pressed: X=279, Y=320, delta=-120
   CHARTEVENT_MOUSE_WHEEL: RightButton pressed: X=253, Y=330, delta=120  */
```

Vedi anche

Funzioni Event Handling, Operazioni con gli eventi

Timeframes del Chart

Tutti i timeframes predefiniti dei charts hanno identificatori univoci. L'identificatore PERIOD_CURRENT significa il periodo attuale del chart su cui è in esecuzione il programma-mql5.

ENUM_TIMEFRAMES

ID	Descrizione
PERIOD_CURRENT	Timeframe corrente
PERIOD_M1	1 minuto
PERIOD_M2	2 minuti
PERIOD_M3	3 minuti
PERIOD_M4	4 minuti
PERIOD_M5	5 minuti
PERIOD_M6	6 minuti
PERIOD_M10	10 minuti
PERIOD_M12	12 minuti
PERIOD_M15	15 minuti
PERIOD_M20	20 minuti
PERIOD_M30	30 minuti
PERIOD_H1	1 ora
PERIOD_H2	2 ore
PERIOD_H3	3 ore
PERIOD_H4	4 ore
PERIOD_H6	6 ore
PERIOD_H8	8 ore
PERIOD_H12	12 ore
PERIOD_D1	1 giorno
PERIOD_W1	1 settimana
PERIOD_MN1	1 mese

Esempio:

```
   string chart_name="test_Object_Chart";
   Print("Cerchiamo di creare un oggetto grafico con il nome",chart_name);
//--- Se un tale oggetto non esiste - creiamolo
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

```
//--- Definire il simbolo
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- Imposta la coordinata X del punto di ancoraggio
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- Imposta la coordinata Y del punto di ancoraggio
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- Imposta la larghezza del grafico
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- Imposta l'altezza
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- Impostare il timeframe
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- Imposta la scalatura --- (da 0 a 5)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- Disattiva la selezione con il mouse
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Identificatori timeseries

Gli identificatori di timeseries sono usati nelle funzioni iHighest() e iLowest(). Possono essere uguali ad un valore dell'enumerazione

ENUM_SERIESMODE

Identificatore	Descrizione
MODE_OPEN	Prezzo di apertura
MODE_LOW	Prezzo Low
MODE_HIGH	Prezzo High
MODE_CLOSE	Prezzo Close
MODE_VOLUME	Volume Tick
MODE_REAL_VOLUME	Volume Reale
MODE_SPREAD	Spread

Vedi anche

PeriodSeconds, Periodo, Data ed Ora, Visibilità degli oggetti

Proprietà Grafico Chart

Identificatori di enumerazioni ENUM_CHART_PROPERTY vengono utilizzati come parametri di funzioni per lavorare con i graici. L'abbreviazione di r/o nella colonna "Tipo della Proprietà" significa che questa proprietà è di sola lettura e non può essere modificata. L' abbreviazione w/o nella colonna "Tipo della Proprietà" significa che questa proprietà è di sola scrittura e non può essere ricevuta. Quando si accede a determinate proprietà, è necessario specificare un aggiuntivo parametro-modificatore (modificatore), che serve ad indicare il numero di sottofinestre grafico-chart. 0 significa la finestra principale.

Le funzioni che definiscono le proprietà del chart sono effettivamente utilizzate per l'invio di comandi di cambio al chart. Se queste funzioni vengono eseguite correttamente, il comando viene incluso nella coda comune degli eventi chart. Le modifiche vengono applicate al chart quando si tratta della coda degli eventi del chart.

Quindi, non aspettatevi un aggiornamento visivo immediato dal chart dopo la chiamata di queste funzioni. In generale, il chart viene aggiornato automaticamente dal terminale in seguito agli eventi di cambiamento - l'arrivo di una nuova quotazione, il ridimensionamento della finestra chart, ecc. Usare la funzione ChartRedraw() per aggiornare forzatamente il chart.

Per le funzioni ChartSetInteger() e ChartGetInteger()

ENUM_CHART_PROPERTY_INTEGER

ID	Descrizione	Tipo di proprietà
CHART_SHOW	Disegno chart prezzi, se false, il disegno di tutti gli attributi del grafico chart dei prezzi è disabilitato e tutti i rientri del bordo del chart vengono eliminati, comprese le scale del tempo e dei prezzi, barra di navigazione rapida, etichette degli eventi del calendario, etichette di trade, descrizioni degli indicatori e delle barre, sottofinestre degli indicatori, istogrammi del volume, ecc. Disabilitare il disegno è una soluzione perfetta per creare un'interfaccia di programma personalizzata usando le risorse grafiche. Gli oggetti grafici vengono sempre tracciati indipendentemente dal valore della proprietà CHART_SHOW.	bool
CHART_IS_OBJECT	Identifica l'oggetto "Chart" (OBJ_CHART) - restituisce true per un oggetto chart. Restituisce false per un grafico chart vero e proprio	bool r/o
CHART_BRING_TO_TOP	Vede il grafico chart sopra altri chart	bool

ID	Descrizione	Tipo di proprietà
CHART_MOUSE_SCROLL	Scorre la tabella utilizzando orizzontalmente il tasto sinistro del mouse. Lo scrolling verticale è disponibile anche se il valore di ogni seguente proprietà è impostato su true: CHART_SCALEFIX, CHART_SCALEFIX_11 o CHART_SCALE_PT_PER_BAR Quando CHART_MOUSE_SCROLL=false, lo scorrimento del chart con la rotellina del mouse non è disponibile	bool
CHART_EVENT_MOUSE_MOVE	Invia notifiche di movimento del mouse ed eventi click del mouse (CHARTEVENT_MOUSE_MOVE) a tutti i programmi MQL5 programmi sul chart	bool
CHART_EVENT_OBJECT_CREATE	Invia la notifica di un evento di creazione del nuovo oggetto (CHARTEVENT_OBJECT_CREATE) a tutti i programmi mql5 sul chart	bool
CHART_EVENT_OBJECT_DELETE	Invia la notifica di un evento di eliminazione oggetto (CHARTEVENT_OBJECT_DELETE) a tutti i programmi mql5 sul chart	bool
CHART_MODE	Tipo di grafico-chart (a candele, a barre o a linea)	enum ENUM_CHART_MODE
CHART_FOREGROUND	Chart dei prezzi in primo piano	bool
CHART_SHIFT	Modalità di chart dei prezzi indentata dal bordo destro	bool
CHART_AUTOSCROLL	Modalità automatica di spostamento al bordo destro del chart	bool
CHART_KEYBOARD_CONTROL	Consenti la gestione del chart utilizzando una tastiera ("Home", "Fine", "Pagina su", "+", "-", "Freccia su", ecc.). Impostando CHART_KEYBOARD_CONTROL su false disabilita lo scorrimento e il ridimensionamento del chart lasciando intatta la possibilità di ricevere i tasti premendo gli eventi in OnChartEvent().	bool
CHART_QUICK_NAVIGATION	Permette al chart di intercettare, quando si preme Spazio o Invio, per attivare la barra di navigazione veloce. La barra di navigazione veloce appare automaticamente nella parte inferiore del chart dopo aver fatto doppio clic sul mouse	bool

ID	Descrizione	Tipo di proprietà
	o premendo Spazio/Invio. Essa consente di modificare rapidamente un simbolo, un timeframe e la data della prima barra visibile.	
CHART_SCALE	Scala	int da 0 a 5
CHART_SCALEFIX	Modalità scala fissa	bool
CHART_SCALEFIX_11	Modalità Scala 1:1	bool
CHART_SCALE_PT_PER_BAR	Scala da specificare in punti per barre	bool
CHART_SHOW_TICKER	Visualizza un simbolo ticker nell'angolo in alto a sinistra. L'impostazione di CHART_SHOW_TICKER su 'false' imposta anche CHART_SHOW_OHLC su "false" e disabilita l' OHLC	bool
CHART_SHOW_OHLC	Visualizza i valori OHLC nell'angolo in alto a sinistra. L'impostazione di CHART_SHOW_OHLC su 'true' imposta anche CHART_SHOW_TICKER su "true" ed abilita il ticker	bool
CHART_SHOW_BID_LINE	Mostra i valori Bid come una linea orizzontale nel chart	bool
CHART_SHOW_ASK_LINE	Mostra valori Ask come una linea orizzontale nel chart	bool
CHART_SHOW_LAST_LINE	Visualizza valori Last come una linea orizzontale nel chart	bool
CHART_SHOW_PERIOD_SEP	Visualizza separatori verticali tra periodi adiacenti	bool
CHART_SHOW_GRID	Visualizza griglia nel chart	bool
CHART_SHOW_VOLUMES	Visualizza volume nel chart	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	Descrizioni Pop-up di oggetti grafici	bool
CHART_VISIBLE_BARS	Il numero di barre sul chart che può essere visualizzato	int r/o
CHART_WINDOWS_TOTAL	Il numero totale di finestre del chart, tra cui sottofinestre indicatore	int r/o
CHART_WINDOW_IS_VISIBLE	Visibilità delle sottofinestre	bool r/o modificatore - numero sottofinestra

ID	Descrizione	Tipo di proprietà
CHART_WINDOW_HANDLE	Handle della finestra del chart (HWND)	int r/o
CHART_WINDOW_YDISTANCE	La distanza tra il frame superiore della sottofinestra grafico-indicatore ed il frame superiore della finestra grafico-chart principale, lungo l'asse verticale Y, in pixel. In caso di un evento del mouse, le coordinate del cursore vengono passate in termini di coordinate della finestra grafico-principale, mentre le coordinate di oggetti grafici in una finestra indicatore secondaria vengono impostati rispetto all'angolo in alto a sinistra della finestra secondaria. Il valore è necessario per convertire le coordinate assolute del grafico principale alle coordinate locali della sottofinestra per il corretto lavoro con gli oggetti grafici, le cui coordinate sono impostate rispetto all'angolo superiore sinistro del frame sottofinestra.	int r/o modificatore - numero sottofinestra
CHART_FIRST_VISIBLE_BAR	Numero della prima barra visibile nel grafico chart. L'indicizzazione delle barre è uguale per TimeSeries.	int r/o
CHART_WIDTH_IN_BARS	Larghezza del chart in barre	int r/o
CHART_WIDTH_IN_PIXELS	Larghezza chart in pixel	int r/o
CHART_HEIGHT_IN_PIXELS	Altezza del chart in pixel	int modifier - numero sottofinestra
CHART_COLOR_BACKGROUND	Colore di sfondo del Chart	color
CHART_COLOR_FOREGROUND	Colore degli assi, scale e linea OHLC	color
CHART_COLOR_GRID	Colore della griglia	color
CHART_COLOR_VOLUME	Colore dei volumi e livelli di apertura della posizione	color
CHART_COLOR_CHART_UP	Colore per la barra in alto, ombre bordi del corpo di candele bull	color
CHART_COLOR_CHART_DOWN	Colore per la barra in alto, ombre bordi del corpo di candele bear	color

ID	Descrizione	Tipo di proprietà
CHART_COLOR_CHART_LINE	Colore linea chart e colore della candela giapponese "Doji"	color
CHART_COLOR_CANDLE_BULL	Colore del corpo della candela toro	color
CHART_COLOR_CANDLE_BEAR	Colore del corpo della candela bear	color
CHART_COLOR_BID	Colore del livello di prezzo Bid	color
CHART_COLOR_ASK	Colore del livello di prezzo Ask	color
CHART_COLOR_LAST	Colore della linea dell'ultimo prezzo affare eseguito (Last)	color
CHART_COLOR_STOP_LEVEL	Colore dei livelli di ordini di stop (Stop Loss e Take Profit)	color
CHART_SHOW_TRADE_LEVELS	Visualizzazione dei livelli di trade nel grafico-chart (livelli di posizioni Aperte, Stop Loss, Take Profit ed Ordini Pendenti)	bool
CHART_DRAG_TRADE_LEVELS	Il permesso di trascinare i livelli di trading sul chart con il mouse. La modalità di trascinamento è attivata per impostazione predefinita (valore true)	bool
CHART_SHOW_DATE_SCALE	Risultati della scala temporale sul chart	bool
CHART_SHOW_PRICE_SCALE	Risultati della scala dei prezzi sul chart	bool
CHART_SHOW_ONE_CLICK	Mostra il pannello "One click trading"sul chart	bool
CHART_SHOW_TRADE_HISTORY	Mostra i trade dalla cronologia di trading come frecce di entrata/uscita su un grafico. Vedi le descrizioni delle opzioni "Mostra cronologia di trading" nelle impostazioni della piattaforma.	bool
CHART_IS_MAXIMIZED	La finestra del chart è massimizzata	bool r/o
CHART_IS_MINIMIZED	La finestra del chart è minimizzata	bool r/o
CHART_IS_DOCKED	La finestra del chart è ancorata. Se impostato su false, il grafico-chart può essere spostato, trascinato fuori dall'area del terminale	bool
CHART_FLOAT_LEFT	La coordinata sinistra della finestra del grafico-chart disancorata rispetto allo schermo virtuale	int

ID	Descrizione	Tipo di proprietà
CHART_FLOAT_TOP	La coordinata superiore della finestra del grafico-chart disancorata rispetto allo schermo virtuale	int
CHART_FLOAT_RIGHT	La coordinata destra della finestra del grafico-chart disancorata rispetto allo schermo virtuale	int
CHART_FLOAT_BOTTOM	La coordinata inferiore della finestra del grafico disancorata rispetto allo schermo virtuale	int

Per le funzioni ChartSetDouble() e ChartGetDouble()

ENUM_CHART_PROPERTY_DOUBLE

ID	Descrizione	Tipo di proprietà
CHART_SHIFT_SIZE	La grandezza della barra zero indentata dal bordo destro, in percentuale	double (da 10 a 50 percento)
CHART_FIXED_POSITION	Posizione fissa dal bordo sinistro in valore percentuale. La posizione fissa del chart è contrassegnata da un piccolo triangolo grigio sull'asse del tempo orizzontale. Viene visualizzato solo se lo slittamento automatico del chart a destra sul segno di spunta in entrata è disattivato (vedere la proprietà CHART_AUTOSCROLL). La barra in una posizione fissa rimane nello stesso posto	double

ID	Descrizione	Tipo di proprietà
	durante lo zoom in e out.	
CHART_FIXED_MAX	Massimo fissato del chart	double
CHART_FIXED_MIN	Minimo fissato del chart	double
CHART_POINTS_PER_BAR	Scala in punti per barra	double
CHART_PRICE_MIN	Minimo del chart	double r/o modificatore - numero sottofinestra
CHART_PRICE_MAX	Massimo del chart	double r/o modificatore - numero sottofinestra

Per le funzioni ChartSetString() e ChartGetString()

ENUM_CHART_PROPERTY_STRING

ID	Descrizione	Tipo di proprietà
CHART_COMMENT	Commento di testo in un chart	string
CHART_EXPERT_NAME	Il nome dell'Expert Advisor in esecuzione sul grafico con il chart_id specificato	string r/o
CHART_SCRIPT_NAME	Il nome dello script in esecuzione sul grafico con il chart_id specificato	string r/o

Esempio:

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
  {
   case(CHART_BARS):    Print("CHART_BARS");   break;
   case(CHART_CANDLES): Print("CHART_CANDLES");break;
   default:Print("CHART_LINE");
  }
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
```

```
   else Print("CHART_SHIFT = false");
   bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
   if(autoscroll) Print("CHART_AUTOSCROLL = true");
   else Print("CHART_AUTOSCROLL = false");
   int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
   Print("CHART_WINDOW_HANDLE = ",chartHandle);
   int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("CHART_WINDOWS_TOTAL = ",windows);
   if(windows>1)
     {
      for(int i=0;i<windows;i++)
        {
         int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
         double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
         double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
         Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
         Print(i+": CHART_PRICE_MIN = ",priceMin);
         Print(i+": CHART_PRICE_MAX = ",priceMax);
        }
     }
```

Vedi anche

Esempi di Lavoro con il Chart

Posizionamento Costanti

Tre identificatori dall'elenco ENUM_CHART_POSITION sono i possibili valori del parametro *position* per la funzione ChartNavigate().

ENUM_CHART_POSITION

ID	Descrizione
CHART_BEGIN	Inizio del Grafico (i prezzi più vecchi)
CHART_CURRENT_POS	Posizione attuale
CHART_END	Fine del Grafico (gli ultimi prezzi)

Esempio:

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
  {
   ChartSetInteger(handle,CHART_AUTOSCROLL,false);
   ChartSetInteger(handle,CHART_SHIFT,true);
   ChartSetInteger(handle,CHART_MODE,CHART_LINE);
   ResetLastError();
   bool res=ChartNavigate(handle,CHART_END,150);
   if(!res) Print("Navigazione fallita. Error = ",GetLastError());
   ChartRedraw();
  }
```

Rappresentazione Grafico

I prezzi dei grafici possono essere visualizzati in tre modi:

- come barre;
- come candele;
- come una linea.

Il modo specifico di visualizzare il grafico dei prezzi è impostato dalla funzione ChartSetInteger(Chart_handle,CHART_MODE, Chart_mode), dove chart_mode è uno dei valori dell'enumerazione ENUM_CHART_MODE.

ENUM_CHART_MODE

ID	Descrizione
CHART_BARS	Visualizzare come una sequenza di barre
CHART_CANDLES	Visualizza come candele giapponesi
CHART_LINE	Visualizza come una linea tracciata dai prezzi Close

Per specificare la modalità di visualizzazione dei volumi nel grafico dei prezzi la funzione ChartSetInteger(Chart_handle, CHART_SHOW_VOLUMES, Volume_mode) viene utilizzata, quando volume_mode è uno dei valori dell'enumerazione ENUM_CHART_VOLUME_MODE.

ENUM_CHART_VOLUME_MODE

ID	Descrizione
CHART_VOLUME_HIDE	I volumi non sono mostrati
CHART_VOLUME_TICK	Volumi Tick
CHART_VOLUME_REAL	Volumi Trade

Esempio:

```
//--- Prende l'handle del grafico corrente
   long handle=ChartID();
   if(handle>0) // se è riuscito, personalizza addizionalmente
     {
      //--- Disabilita l' autoscroll
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- Imposta l'indendazione del borde destro del grafico
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- Mostra come candele
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- Scorrere di 100 barre a partire dall'inizio dello storico
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- Imposta la modalità di visualizzazione volume tick
```

```
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
     }
```

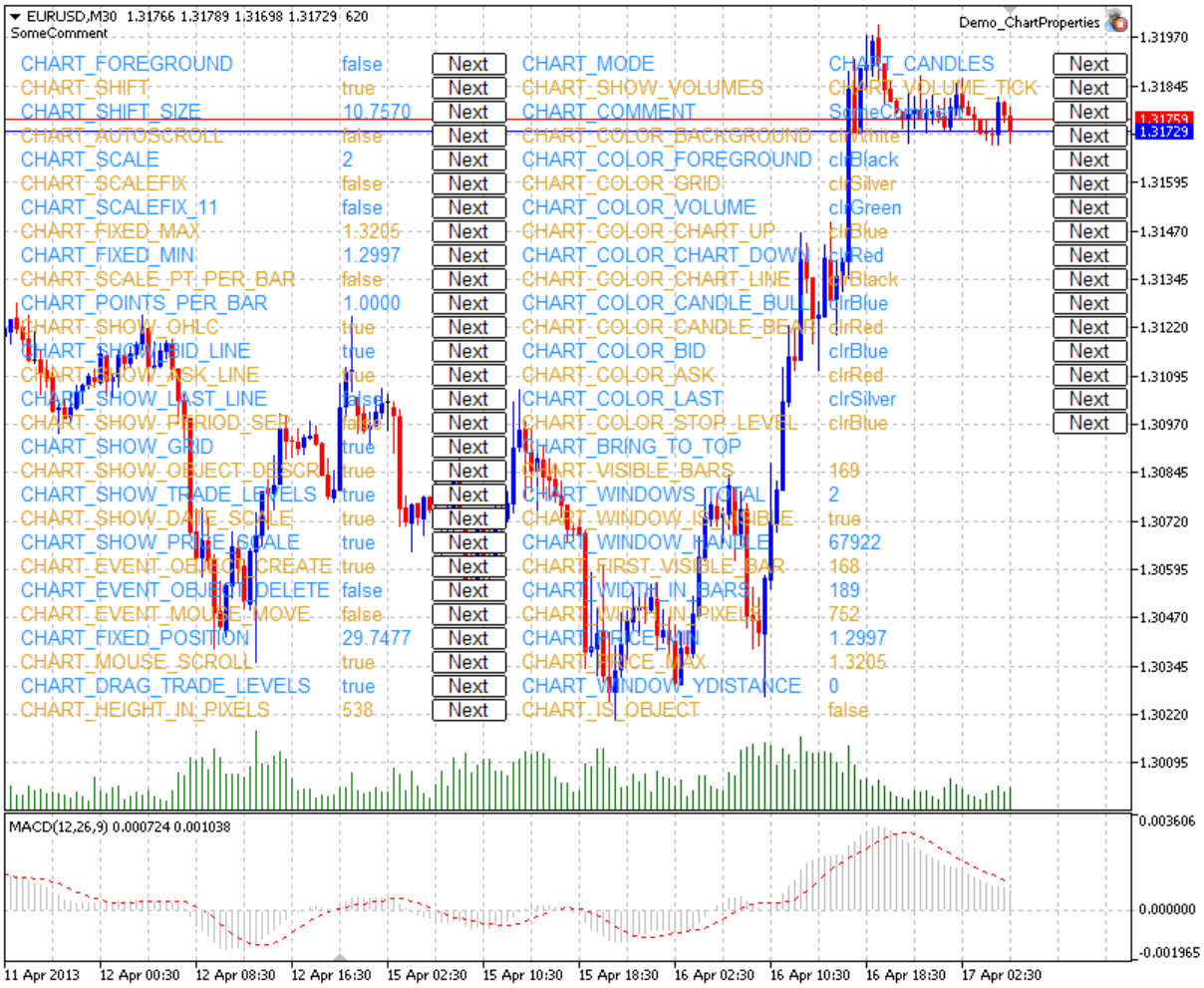
Vedi anche

ChartOpen, ChartID

Esempi di Lavoro con il grafico Chart

Questa sezione contiene esempi di lavoro con le proprietà del chart. Uno o due funzioni complete vengono visualizzate per ogni proprietà. Queste funzioni permettono di impostare/ricevere il valore della proprietà. Queste funzioni possono essere utilizzate "così come sono" nelle applicazioni mql5 personalizzate.

La figura seguente mostra il pannello grafico che illustra come la modifica delle proprietà del chart cambia il suo aspetto. Il click sul pulsante Avanti permette di impostare il nuovo valore della proprietà appropriata e visualizzare le modifiche nella finestra del grafico.



Il codice sorgente del pannello si trova sotto.

Proprietà del Chart e Funzioni di Esempio per lavorare con Esse

- **CHART_IS_OBJECT**definisce se un oggetto è un chart reale o un oggetto grafico.

```
//+------------------------------------------------------------------------------+
//| Controlla se un oggetto è un chart. Se è un oggetto grafico,                 |
//| il risultato è true. Se è un vero chart, la variable risultato               |
//| ha il valore di false.                                                       |
//+------------------------------------------------------------------------------+
bool ChartIsObject(bool &result,constlong chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
/ --- ottiene la proprietà chart
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      //--- restituisce false
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_BRING_TO_TOP** mostra il chart in cima a tutti gli altri.

```
//+------------------------------------------------------------------------------+
//| Invia il comando al terminale per visualizzare il chart sopra tutti gli altri |
//+------------------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- mostra il chart in cima a tutti gli altri
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_MOUSE_SCROLL** è una proprietà di scorrimento del chart usando il tasto sinistro del mouse.

```
//+------------------------------------------------------------------------------------+
//| Controlla se lo scorrimento del chart usando il tasto sinistro del mouse è abilita
//+------------------------------------------------------------------------------------+
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita lo scorrimento del chart usando il bottone sinistro del mouse
//+------------------------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE** è una proprietà di invio messaggi riguardanti eventi di movimento e click del mouse per applicazioni mql5 (CHARTEVENT_MOUSE_MOVE).

```
//+------------------------------------------------------------------------------------+
//| Controlla se i messaggi concernenti eventi di spostamento ed i click del mouse |
//| vengono inviati a tutte le applicazioni MQL5 sul chart                          |
//+------------------------------------------------------------------------------------+
```

```
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la modalità invio messaggi concernente eventi di spostamento
//| e di click del mouse alle applicazioni MQL5 sul chart
//+------------------------------------------------------------------------------------+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE** è una proprietà di invio messaggi riguardanti l'evento di creazione di un oggetto grafico in applicazioni mql5 (CHARTEVENT_OBJECT_CREATE).

```
//+------------------------------------------------------------------------------------+
//| Controlla se i messaggi riguardanti gli eventi di
//| creazione degli oggetti grafici vengono inviati a tutte le applicazioni mql5 sul c
//+------------------------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
```

```
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Abilita/disabilita la modalità di invio messaggi riguardanti gli eventi di
//| creazione di un oggetto grafico su tutte le applicazioni mql5 sul chart
//+------------------------------------------------------------------------------------
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE** è una proprietà di trasmissione dei messaggi concernente l'evento di delezione(eliminazione) di un oggetto grafico in applicazioni mql5 (CHARTEVENT_OBJECT_DELETE).

```
//+------------------------------------------------------------------------------------
//| Controlla se i messaggi concernenti l'evento dell'eliminazione
//| di oggetti grafici, viene inviato a tutte le applicazioni mql5 sul chart
//+------------------------------------------------------------------------------------
bool ChartEventObjectDeleteGet(bool &result,constlong chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
```

```
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la modalità di invio messaggi riguardanti gli eventi di |
//| eliminazione di oggetti grafici a tutte le applicazioni mql5 sul chart |
//+------------------------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_MODE** - tipo di chart (candele, barre o linea).

```
//+------------------------------------------------------------------------------------+
//| Ottiene il tipo di display del chart (candele, barre o linea)              |
//+------------------------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=WRONG_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
```

```
   return((ENUM_CHART_MODE)result);
  }
//+------------------------------------------------------------------------------------+
//| Imposta il tipo di display del chart (candele, barre o linea)                      |
//+------------------------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_FOREGROUND** è una proprietà di visualizzazione di un chart dei prezzi in primo piano.

```
//+------------------------------------------------------------------------------------+
//| Controlla se il chart dei prezzi è visualizzato in primo piano |
//+------------------------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione del chart dei prezzi in primo piano   |
//+------------------------------------------------------------------------------------+
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
```

```
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHIFT** - modalità di spostamento del chart dei prezzi dal bordo destro.

```
//+------------------------------------------------------------------------------------
//| Controlla se lo slittamento del chart dei prezzi dal bordo destro è abilitata |
//+------------------------------------------------------------------------------------
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Abilita/disabilita la visualizzazione del chart prezzi con uno shift dal bordo des
//+------------------------------------------------------------------------------------
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

```
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_AUTOSCROLL** - la modalità di slittamento automatico al bordo destro del chart.

```
//+------------------------------------------------------------------------------------+
//| Controlla se lo scorrimento automatico del chart sulla destra                      |
//| sull'arrivo dei nuovi ticks è abilitata                                            |
//+------------------------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita lo scorrimento automatico del chart sulla destra     |
//| all'arrivo di nuovi ticks                                                          |
//+------------------------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

```
```

- **CHART_SCALE** - proprietà di scala del chart.

```
//+------------------------------------------------------------------------------+
//| Ottiene la scala del chart (da 0 a 5)                                        |
//+------------------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta la scala del chart (da 0 a 5)                                        |
//+------------------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SCALEFIX** - la modalità di scala fissa del chart.

```
//+------------------------------------------------------------------------------+
//| Controlla se la modalità scala fissa è abilitata                             |
//+------------------------------------------------------------------------------+
```

```
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Abilita/disabilita la modalità scala fissa                                   |
//+------------------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SCALEFIX_11** - modalità 1:1 di scala del chart.

```
//+------------------------------------------------------------------------------+
//| Controlla se la scala "1:1" è abilitata                                      |
//+------------------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
```

```
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Abilita/disabilita la modalità scala "1:1"                                   |
//+------------------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - la modalità di specificazione della scala del chart, in punti per barra.

```
//+------------------------------------------------------------------------------+
//| Controlla se la modalità di scala "punti per barra" è abilitata |
//+------------------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
```

```
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la modalità di scala del chart "punti per barra" |
//+------------------------------------------------------------------------------------+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_OHLC** - la proprietà di visualizzazione di valori OHLC nell'angolo superiore sinistro.

```
//------------------------------------------------------------------------------------
//| Controlla se la visualizzazione dei valori OHLC nell'angolo superiore sinistro del
//------------------------------------------------------------------------------------
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//------------------------------------------------------------------------------------
//| Abilita/disabilita la visualizzazione dei valori OHLC nell'angolo superiore sinist
//------------------------------------------------------------------------------------
```

```
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - la proprietà di visualizzazione del valore Bid come linea orizzontale sul chart.

```
//+------------------------------------------------------------------------------------+
//| Controlla se la visualizzazione della linea Bid sul chart è abilitata |
//+------------------------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione della linea Bid sul chart                |
//+------------------------------------------------------------------------------------+
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
```

```
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_ASK_LINE** - la proprietà di visualizzazione valori Ask come linea orizzontale su un chart.

```
//+------------------------------------------------------------------------------------+
//| Controlla se la visualizzazione della linea Ask sul chart è abilitata |
//+------------------------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione della linea Ask sul chart |
//+------------------------------------------------------------------------------------+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
```

```
   return(true);
  }
```

- **CHART_SHOW_LAST_LINE** - la proprietà di visualizzazione valore Last come una linea orizzontale in un chart.

```
//+------------------------------------------------------------------------------------
//| Controlla se la visualizzazione della linea per l'ultimo prezzo eseguito, è abilit
//+------------------------------------------------------------------------------------
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Abilita/disabilita la visualizzazione della linea per l'ultimo prezzo eseguito |
//+------------------------------------------------------------------------------------
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_PERIOD_SEP** - la proprietà di visualizzazione dei separatori verticali tra periodi adiacenti.

```
//+------------------------------------------------------------------------------------
//| Controlla se la visualizzazione dei separatori verticali tra periodi adiacenti è a
//+------------------------------------------------------------------------------------
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Abilita/disabilita la visualizzazione dei separatori verticali tra periodi adiacer
//+------------------------------------------------------------------------------------
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_GRID** - la proprietà di visualizzazione di della griglia del chart.

```
//+------------------------------------------------------------------------------+
//| Controlla se la griglia del chart è visualizzata                             |
//+------------------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
```

```
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione della griglia sul chart                      |
//+------------------------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_VOLUMES** - la proprietà di visualizzazione dei volumi sul chart.

```
//+------------------------------------------------------------------------------------+
//| Controlla se i volumi vengono visualizzati sul chart (tre possibilità:             |
//| non vengono visualizzati, volumi tick visualizzati, volumi reali visualizzati)     |
//+------------------------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=WRONG_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
```

```
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta la modalità di visualizzazione dei volumi sul chart                  |
//+------------------------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** - la proprietà della descrizione pop-up dell'oggetto grafico.

```
//+------------------------------------------------------------------------------+
//| Controlla se le descrizioni descriptions degli oggetti grafici vengono visualizzat
//| quando si tiene il mouse su esse                                             |
//+------------------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
```

```
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione di descrizioni pop-up di oggetti grafici |
//| quando si tiene il mouse su esse                                                   |
//+------------------------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_VISIBLE_BARS** definisce il numero di barre su un grafico che sono disponibili per la visualizzazione.

```
//+------------------------------------------------------------------------------+
//| Ottiene il numero di barre che sono visualizzate (visibili) nel chart |
//+------------------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL** definisce il numero totale di finestre chart, tra cui le sottofinestre indicatori.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il numero totale di finestre del chart, tra cui indicatori sottofinestre |
//+------------------------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
```

- **CHART_WINDOW_IS_VISIBLE** definisce la visibilità della sottofinestra.

```
//+------------------------------------------------------------------------------------+
//| Verifica se la finestra del grafico corrente, o sottofinestra, è visibile |
//+------------------------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_WINDOW_HANDLE** restituisce l'handle del chart.

```
//+------------------------------------------------------------------------------+
//| Ottiene l'handle del chart                                                   |
//+------------------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE** definisce la distanza in pixel tra il frame superiore della sottofinestra indicatore ed il frame superiore della finestra principale del chart.

```
//+------------------------------------------------------------------------------+
//| Ottiene la distanza in pixel tra il bordo superiore di          |
//| Sottoffinestra e bordo superiore della finestra superiore del chart |
//+------------------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR** restituisce il numero della prima barra visibile sul chart (indicizzazione barra corrisponde alle timeseries).

```
//--------------------------------------------------------------------------------
//| Ottiene l'indice della prima barra visibile sul chart |
//| L'indicizzazione avviene come in timeseries: le ultime barre hanno indice inferio
//--------------------------------------------------------------------------------
int ChartFirstVisibleBar(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS** restituisce lo spessore del chart, in barre.

```
//+--------------------------------------------------------------------------------+
//| Ottiene lo spessore del chart (in barre)                                       |
//+--------------------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
```

- **CHART_WIDTH_IN_PIXELS** restituisce lo spessore del chart, in pixels.

```
//+--------------------------------------------------------------------------------+
//| Ottiene lo spessore del chart (in pixels)                                      |
//+--------------------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
```

```
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
```

- **CHART_HEIGHT_IN_PIXELS** - proprietà altezza del chart, in pixels.

```
//+------------------------------------------------------------------------------------+
//| Ottiene l'altezza del chart (in pixels)                                            |
//+------------------------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long result=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((int)result);
  }
//+------------------------------------------------------------------------------------+
//| Imposta l'altezza del chart (in pixels)                                            |
//+------------------------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

```
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND -** colore di sfondo del chart.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il colore di sottofondo del chart                                          |
//+------------------------------------------------------------------------------------+
color ChartBackColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore di sfondo del chart
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------------+
//| Imposta il colore di sottofondo del chart                                          |
//+------------------------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore di sfondo del chart
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_FOREGROUND -** colore degli assi, la scala e la linea OHLC.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il colore degli assi, scala e linea OHLC                                   |
```

```
//+------------------------------------------------------------------------------+
color ChartForeColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il colore degli assi, dekla scala e della linea OHLC
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il colore degli assi, scala e linea OHLC                 |
//+------------------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore degli assi, della scala e della linea OHLC
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_GRID** - colore della griglia del chart.

```
//+------------------------------------------------------------------------------+
//| Imposta il colore della griglia del chart                                    |
//+------------------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore della griglia del chart
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
```

```
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------------+
//| Imposta il colore della griglia del chart                                          |
//+------------------------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore della griglia del chart
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_VOLUME** - colore di volumi e dei livelli di apertura posizione.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il colore dei livelli di entry dei volumi e del market |
//+------------------------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore dei volumi e dei livelli di ingresso nel mercato
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------------+
//| Imposta il colore dei livelli di entry dei volumi e del market |
//+------------------------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
```

```
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta i colori dei volumi e dei livelli di ingresso sul mercato
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_CHART_UP** - colore della barra superiore, la sua ombra ed il bordo del corpo della candela rialzista.

```
//+------------------------------------------------------------------------------------------+
//| Ottiene il colore di barra superiore, ombra e bordo (corpo di candela bullish) |
//+------------------------------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore della barra superiore, la sua ombra ed il bordo del corpo della
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------------------+
//| Imposta il colore di barra superiore, ombra e bordo (corpo di candela bullish) |
//+------------------------------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore della barra superiore, la sua ombra ed bordo del corpo di una
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

```
      }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - colore della barra inferiore, la sua ombra ed bordo del corpo della candela ribassista.

```
//+------------------------------------------------------------------------------+
//| Ottiene il colore di barra inferiore, ombra e bordo (corpo di candela bullish) |
//+------------------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore della barra inferiore, la sua ombra ed il bordo del corpo della
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il colore di barra inferiore, ombra e bordo (corpo di candela bullish) |
//+------------------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore della barra inferiore, la sua ombra ed il bordo del corpo del
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - colore della linea del chart e della candela Doji.

```
//+------------------------------------------------------------------------------+
```

```
//| Ottiene il colore della linea del chart e candele Doji |
//+------------------------------------------------------------------------------+
color ChartLineColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore della linea del chart e della candela Doji
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il colore della linea del chart e candele Doji |
//+------------------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta i colori della linea del grafico e delle candele Doji
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** - colore del bordo della candela rialzista.

```
//+------------------------------------------------------------------------------+
//| Ottiene il colore del corpo della candela bullish |
//+------------------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore del corpo della candela rialzista
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
```

```
      {
       //--- visualizza il messaggio di errore nel journal Experts
       Print(__FUNCTION__+", Error Code = ",GetLastError());
      }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il colore del corpo della candela bullish |
//+------------------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore del corpo della candela rialzista
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** - colore del corpo della candela ribassista.

```
//+------------------------------------------------------------------------------+
//| Ottiene il colore del corpo della candela bearish |
//+------------------------------------------------------------------------------+
color ChartBearColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore del corpo della candela ribassista
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il colore del corpo della candela bearish |
//+------------------------------------------------------------------------------+
```

```
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore del corpo della candela ribassista
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_BID** - colore della linea di prezzo Bid.

```
//+------------------------------------------------------------------------------+
//| Imposta il colore della linea Bid                                            |
//+------------------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore della linea di prezzo Bid
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il colore della linea Bid                                            |
//+------------------------------------------------------------------------------+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore della linea di prezzo Bid
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

```
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_ASK** - colore della linea di prezzo Ask.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il colore della linea Ask                                                  |
//+------------------------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore della linea di prezzo Ask
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------------+
//| Imposta il colore della linea Ask                                                  |
//+------------------------------------------------------------------------------------+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore della linea di prezzo Ask
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_LAST** - colore della linea di prezzo dell'ultimo affare eseuito (Last).

```
//+------------------------------------------------------------------------------------+
//| Ottiene il colore della linea dell'ultimo affare eseguito |
```

```
//+------------------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il colore della lina di prezzo dell'ultimo affare eseguito (Last)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il colore della linea dell'ultimo affare eseguito |
//+------------------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore della lina di prezzo dell'ultimo affare eseguito (Last)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - colore dei livelli di stop order (Stop Loss e Take Profit).

```
//+------------------------------------------------------------------------------+
//| Ottiene il colore dei livelli di Stop Loss e Take Profit                     |
//+------------------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- prepara la variabile a ricevere il colore
   long result=clrNONE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il colore dei livelli degli stop orders (Stop Loss e Take Profit)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
```

```
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return((color)result);
  }
//+------------------------------------------------------------------------------------+
//| Imposta il colore dei livelli di Stop Loss e Take Profit                           |
//+------------------------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il colore dei livelli degli ordini di stop (Stop Loss e Take Profit)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - proprietà di visualizzazione di livelli di trade sul chart (livelli di posizioni aperte, Stop Loss, Take Profit ed ordini in attesa).

```
//+------------------------------------------------------------------------------------+
//| Controlla se i livelli di trading vengono visualizzati sul chart |
//+------------------------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
```

```
//+------------------------------------------------------------------------------+
//| Abilita/Disabilita la visualizzazione dei livelli di trading |
//+------------------------------------------------------------------------------+
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - di proprietà di abilitazione della possibilità di trascinare i livelli di trading su un chart usando il mouse.

```
//+------------------------------------------------------------------------------+
//| Controlla se consente il trascinamento livelli di trading sul chart, col mouse |
//+------------------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Abilita/disabilita il trascinamento dei livelli di trading sul chart col mouse |
//+------------------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
```

```
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_DATE_SCALE** - proprietà di visualizzazione della scala temporale su un chart.

```
//+------------------------------------------------------------------------------------+
//| Controlla se la scala temporale è visualizzata sul chart |
//+------------------------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione della scala temporale sul chart |
//+------------------------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

```
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - di proprietà di visualizzazione della scala di prezzo su un chart.

```
//+------------------------------------------------------------------------------------+
//| ChControlla se la scala prezzo è visualizzata sul chart |
//+------------------------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
/ --- preparara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà chart in memoria
   result=value;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione della scala prezzo sul chart |
//+------------------------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_SHOW_ONE_CLICK** - property of displaying the "One click trading" panel on a chart.

```
//+------------------------------------------------------------------+
// | Controlla se il pannello "One click trading" viene visualizzato sul chart |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- prepara la variabile per ottenere il valore della proprietà
   long value;
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- mostra il messaggio d'errore nel journal dell'Expert
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà del chart nella memoria
   result=value;
//--- esecuzione con successo
   return(true);
  }
//+------------------------------------------------------------------+
//| Abilita/disabilita la visualizzazione del pannello "One click trading" |
//| sul chart                                                        |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- mostra il messaggio d'errore nel journal dell'Expert
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione con successo
   return(true);
  }
```

- **CHART_SHIFT_SIZE** - grandezza di slittamento della barra zero dal bordo destro in valori percentuali.

```
//+------------------------------------------------------------------+
//| Ottiene la grandezza dello slittamento della barra zero dal bordo sinistro |
//| del chart, in valori percentuali (dal 10% fino al 50%) |
```

```
//+------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- prepara la variabile per ottenere il risultato
   double result=EMPTY_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return(result);
  }
//+-----------------------------------------------------------------------------------
//| Ottiene la grandezza dello slittamento della barra zero dal bordo destro |
//| del grafico in valori percentuali (dal 10% fino al 50%).                  |
//| Per abilitare shift mode il valore della proprietà CHART_SHIFT dev'essere true| /
//+-----------------------------------------------------------------------------------
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_FIXED_POSITION** - la posizione fissata del chart dal bordo sinistro in valore percentuale.

```
//+-----------------------------------------------------------------------------------
//| Ottiene il punto della posizione fissa del chart dal bordo sin. (in valori percent
//+-----------------------------------------------------------------------------------
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- prepara la variabile per ottenere il risultato
   double result=EMPTY_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
```

```
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return(result);
  }
//+------------------------------------------------------------------------------------
//| Ottiene il punto della posizione fissa del chart dal bordo sin. (in valori percent
//| Per visualizzare il punto della posizione fissa del chart, il valore della proprie
//| CHART_AUTOSCROLL dev'essere impostato come false  |
//+------------------------------------------------------------------------------------
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_FIXED_MAX** - proprietà del massimo fissato del chart.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il valore del massimo fissato del chart |
//+------------------------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- prepara la variabile per ottenere il risultato
   double result=EMPTY_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return(result);
  }
```

```
//+------------------------------------------------------------------------------+
//| Imposta il valore del massimo fissato del chart |                           |
//| Per cambiare il valore della proprietà, il valore della proprietà |
//| dev'essere preliminarmente impostato come true | |
//+------------------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_FIXED_MIN** - proprietà del minimo fissato del chart.

```
//+------------------------------------------------------------------------------+
//| Ottiene il valore del minimo fissato del chart |
//+------------------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- prepara la variabile per ottenere il risultato
   double result=EMPTY_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return(result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il valore del minimo fissato del chart |                           |
//| Per cambiare il valore della proprietà, il valore della proprietà |
//| dev'essere preliminarmente impostato come true | |
//+------------------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
```

```
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** - valore della scala in punti per barra.

```
//+------------------------------------------------------------------------------+
//| Ottiene il valore della scala del chart in punti per barra |
//+------------------------------------------------------------------------------+
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- prepara la variabile per ottenere il risultato
   double result=EMPTY_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return(result);
  }
//+------------------------------------------------------------------------------+
//| Imposta il valore della scala del chart in punti per barra |
//| Per vedere il risultato di questo cambio di valore della proprietà, il valore |
//| CHART_SCALE_PT_PER_BAR deve essere preliminarmente impostato su true. //| |
//+------------------------------------------------------------------------------+
bool (const double value,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
```

```
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_PRICE_MIN** restituisce il valore del minimo del chart.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il valore minimo del chart nella finestra principale (o sottofinestra) |
//+------------------------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- prepara la variabile per ottenere il risultato
   double result=EMPTY_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return(result);
  }
```

- **CHART_PRICE_MAX** resituisce il valore del massimo del chart.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il valore massimo del chart nella finestra principale (o sottofinestra)|
//+------------------------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- prepara la variabile per ottenere il risultato
   double result=EMPTY_VALUE;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- restituisce il valore della proprietà chart
   return(result);
  }
```

- **CHART_COMMENT** - commento sul chart.

```
//+------------------------------------------------------------------------------------+
//| Ottiene il commento nell'angolo superiore sinitro del chart |
//+------------------------------------------------------------------------------------+
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- ricevere il valore della proprietà
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Ottiene il commento nell'angolo superiore sinitro del chart |
//+------------------------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il valore della proprietà
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- visualizza il messaggio di errore nel journal Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

- **CHART_IS_MAXIMIZED** - la finestra del chart è massimizzata.

```
//+------------------------------------------------------------------+
//| Definisce se la finestra corrente del chart è massimizzata       |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- prepara la variabile per ricevere il valore della proprietà
   long value;
//--- resetta il valore dell'errore
   ResetLastError();
//--- riceve il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- mostra un messaggio d'errore nel log degli Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà del chart nella variabile
   result=value;
//--- esecuzione con successo
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - la finestra chart è minimizzata.

```
//+------------------------------------------------------------------+
//| Definisce se la finestra corrente del chart è massimizzata       |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- prepara la variabile per ricevere il valore della proprietà
   long value;
//--- resetta il valore dell'errore
   ResetLastError();
//--- riceve il valore della proprietà
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- mostra l'errore del messaggio nel log dell'Expert
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizza il valore della proprietà del chart nella variabile
   result=value;
//--- esecuzione con successo
   return(true);
  }
```

Pannello per le proprietà del chart

```
//--- connette la libreria di elementi di controllo
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- costanti predefinite
#define X_PROPERTY_NAME_1    10  // coordinate x del nome della proprietà nella prima
#define X_PROPERTY_VALUE_1   225 // coordinate x del valore della proprietà nella prir
```

```
#define X_PROPERTY_NAME_2    345 // coordinate x del nome della proprietà nella second
#define X_PROPERTY_VALUE_2   550 // coordinate x del valore della proprietà nella seco
#define X_BUTTON_1           285 // coordinate x del bottone nella prima colonna
#define X_BUTTON_2           700 // coordinate x del bottone nella seconda colonna
#define Y_PROPERTY_1          30  // coordinate y dell'inizio della prima e seconda col
#define Y_PROPERTY_2         286 // coordinate y dell'inizion della terza colonna
#define Y_DISTANCE            16  // distanza assiale y tra le linee
#define LAST_PROPERTY_NUMBER 111 // numero dell'ultima proprietà grafica
//--- parametri di input
input color InpFirstColor=clrDodgerBlue; // Colore delle righe dispari
input color InpSecondColor=clrGoldenrod; // Colore delle righe pari
//--- variabili ed arrays
CChartObjectLabel  ExtLabelsName[];  // etichetta per visualizzazione dei nomi delle p
CChartObjectLabel  ExtLabelsValue[]; // etichetta per visualizzazione dei valori delle
CChartObjectButton ExtButtons[];     // bottoni
int                ExtNumbers[];     // indici delle proprietà
string             ExtNames[];       // nomi delle proprietà
uchar              ExtDataTypes[];   // tipi di dato delle proprietà (integer, double,
uint               ExtGroupTypes[];  // array che memorizza i dati sulla appartenenza
uchar              ExtDrawTypes[];   // array che memorizza i dati sul tipo di proprie
double             ExtMaxValue[];    // massimo valori delle proprietà che sono possib
double             ExtMinValue[];    // minimi valori delle proprietà che sono possibi
double             ExtStep[];        // steps per il cambio delle proprietà
int                ExtCount;         // numero totale di tutte le proprietà
color              ExtColors[2];     // array di colori per la visualizzazione delle r
string             ExtComments[2];   // array di commenti (per la priprietà CHART_COMM
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato  |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- visualizza un commento sul grafico
   Comment("SomeComment");
//--- memorizza i colori nell'array per potervi passare tra essi in un secondo momento
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- memorizza i commenti nell'array per potervi passare tra essi in un secondo momen
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- prepara e visualizza il pannello di controllo per la gestione delle proprietà de
   if(!PrepareControls())
      return(INIT_FAILED);
//--- esecuzione avvenuta
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione dell'expert |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
```

```
  {
//--- rimuove il commento sul chart
   Comment("");
  }
//+------------------------------------------------------------------------------+
//| Handler di un evento chart |
//+------------------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- controlla l'evento del clic sull'oggetto chart
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- divide il nome dell'oggetto per separatore
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- controlla se l'oggetto è un bottone
      if(obj_name[0]=="Button")
        {
         //--- riceve l'indice del bottone
         int index=(int)StringToInteger(obj_name[1]);
         //--- rilascia il bottone
         ExtButtons[index].State(false);
         //--- imposta il nuovo valore della proprietà a seconda del suo tipo
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- ri-disegna i valori delle proprietà
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------------------+
//| Cambia la proprietà integer del chart                                        |
//+------------------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- riceve il valore corrente della proprietà
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- definisce il seguente valore della proprietà
   switch(ExtDrawTypes[index])
     {
      case 'C':
```

```
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- imposta il nuovo valore della proprietà
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------------------+
//| Cambia la proprietà double del chart |
//+------------------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- riceve il valore corrente della proprietà
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- definisce il seguente valore della proprietà
   value=GetNextValue(value,index);
//--- imposta il nuovo valore della proprietà
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------------------+
//| Cambia la proprietà string del chart |
//+------------------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- variabile statica per il passaggio all'interno dell'array ExtComments
   static uint comment_index=1;
//--- indice di cambiamento per ricevere un altro commento
   comment_index=1-comment_index;
//--- imposta il nuovo valore della proprietà
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+------------------------------------------------------------------------------+
//| Ottiene il prossimo valore della proprietà |
//+------------------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------------------+
//| Ottiene il prossimo colore per la proprietà di tipo colore       |
//+------------------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
//--- restituisce i seguenti valori del colore
```

```
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------------------+
//| Re-disegna i valori della proprietà                                          |
//+------------------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- testo valore della proprietà
   string text;
   long   value;
//--- loop del numero delle proprietà
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
        {
         case 'I':
            //--- riceve il corrente valore della proprietà
            if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
            break;
            //--- testo integer della proprietà
            switch(ExtDrawTypes[i])
              {
               //--- proprietà colore
               case 'C':
                  text=(string)((color)value);
                  break;
                  //--- proprietà booleana
               case 'B':
                  text=(string)((bool)value);
                  break;
                  //--- proprietà di enumerazione ENUM_CHART_MODE
               case 'M':
                  text=EnumToString((ENUM_CHART_MODE)value);
                  break;
                  //--- proprietà di enumerazione ENUM_CHART_VOLUME_MODE
               case 'V':
                  text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                  break;
                  //--- numero tipo int
               default:
                  text=IntegerToString(value);
```

```
                  break;
               }
            break;
         case 'D':
            //--- testo proprietà double
            text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
            break;
         case 'S':
            //--- testo proprietà string
            text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
            break;
        }
      //--- valore della prorpietà display
      ExtLabelsValue[i].Description(text);
     }
  }
//+------------------------------------------------------------------------------+
//| Crea il pannello per gestire le proprietà del chart |
//+------------------------------------------------------------------------------+
bool PrepareControls(void)
  {
//--- alloca la memoria per array con una riseva
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- variabili
   int i=0;     // variable loop
   int col_1=0; // numero di proprietà nella prima colonna
   int col_2=0; // numero di proprietà nella seconda colonna
   int col_3=0; // numero di proprietà nella terza colonna
//--- numero corrente della proprità - 0
   ExtCount=0;
//--- visualizzazione per le proprietà nel loop
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- immagazzina il corrente numero della proprietà
      ExtNumbers[ExtCount]=i;
      //--- incrementa il valore della variabile loop
      i++;
      //--- controlla se c'è una proprietà con tale numero
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- crea elementi di controllo per la proprietà
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- crea etichette d un bottone per la proprietà
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- il numero degli elementi nella prima colonna è aumentato
               col_1++;
```

```
               break;
            case 2:
               //--- crea etichette d un bottone per la proprietà
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- il numero degli elementi nella seconda colonna è aumentato
               col_2++;
               break;
            case 3:
               //--- crea solo etichette per la proprietà
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PI
               return(false);
               //--- il numero degli elementi nella terza colonna è aumentato
               col_3++;
               break;
           }
         //--- definisce il valore della proprietà massimo e minimo e lo step
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
         //--- incrementa il numero delle proprietà
         ExtCount++;
        }
     }
//--- libera la memoria non usata dagli array
   MemoryAllocation(ExtCount);
//--- ri-disegna i valori delle proprietà
   RedrawProperties();
   ChartRedraw();
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Alloca la memoria per gli array |
//+------------------------------------------------------------------------------------+
void MemoryAllocation(const int size)
  {
   ArrayResize(ExtLabelsName,size);
   ArrayResize(ExtLabelsValue,size);
   ArrayResize(ExtButtons,size);
   ArrayResize(ExtNumbers,size);
   ArrayResize(ExtNames,size);
   ArrayResize(ExtDataTypes,size);
   ArrayResize(ExtGroupTypes,size);
   ArrayResize(ExtDrawTypes,size);
   ArrayResize(ExtMaxValue,size);
   ArrayResize(ExtMinValue,size);
   ArrayResize(ExtStep,size);
  }
//+------------------------------------------------------------------------------------+
//| Controlla se l'indice della proprietà appartiene ad uno di quelli dell' |
```

```
//| enumerazioni ENUM_CHART_PROPERTIES                                |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- controllare se la proprietà è di tipo integer
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                       // proprietà dall'enumerazione ENUM_CHART_PR
      GetTypes(ind,group_type,draw_type); // definisce i parametri della proprietà dis
      return(true);
     }
//--- controllare se la proprietà è di tipo double
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                       // proprietà dall' enumerazione ENUM_CHART_P
      GetTypes(ind,group_type,draw_type); // definisce i parametri della proprietà dis
      return(true);
     }
//--- controlla se la proprietà è di tipo stringa
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                       // proprietà dall'enumerazione ENUM_CHART_PR
      GetTypes(ind,group_type,draw_type); // definisce i parametri della proprietà dis
      return(true);
     }
//--- la proprietà non corrisponde a nessuna enumerazione
   return(false);
  }
//+------------------------------------------------------------------+
//| Definisce il gruppo nel quale la proprietà dev'essere memorizzata |
//| così come il tipo della sua visualizzazione |
//+------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- controlla se la proprietà appartiene al terzo gruppo
//--- le proprietà del terzo gruppo vengono visualizzate nella seconda colonna a parti
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- controlla se la proprietà appartiene al secondo gruppo
//--- le proprietà del secondo gruppo sono visualizzate all'inizio della seconda colon
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//---se vi trovate qui, la proprietà appartiene al primo gruppo (prima colonna)
```

```
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------------------+
//| Controlla se la proprietà appartiene al terzo gruppo e |
//| definisce il suo tipo di visualizzazione in caso di risposta positiva |
//+------------------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- controlla se la proprietà appartiene al terzo gruppo
   switch(property_number)
     {
      //--- proprietà booleane
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- proprietà integer
      case CHART_VISIBLE_BARS:
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
      case CHART_WIDTH_IN_BARS:
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- proprietà double
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- in realtà, questa proprietà è un comando di visualizzazione del chart i
         //--- non vi è alcuna necessità di applicare questo pannello, siccome la fine
         //--- in cima ad altre prima che noi la usiamo
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- la proprietà non appartiene al terzo gruppo
      default:
         return(false);
     }
//--- la proprietà appartiene al terzo gruppo
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla se la proprietà appartiene al secondo gruppo e |
//| definisce il suo tipo di visualizzazione in caso di risposta positiva |
//+------------------------------------------------------------------------------+
```

```
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- controlla se la proprietà appartiene al secondo gruppo
   switch(property_number)
     {
      //--- ENUM_CHART_MODE type property
      case CHART_MODE:
         draw_type='M';
         break;
         //--- ENUM_CHART_VOLUME_MODE type property
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- proprietà stringa
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- proprietà color
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
      case CHART_COLOR_CHART_UP:
      case CHART_COLOR_CHART_DOWN:
      case CHART_COLOR_CHART_LINE:
      case CHART_COLOR_CANDLE_BULL:
      case CHART_COLOR_CANDLE_BEAR:
      case CHART_COLOR_BID:
      case CHART_COLOR_ASK:
      case CHART_COLOR_LAST:
      case CHART_COLOR_STOP_LEVEL:
         draw_type='C';
         break;
         //--- la proprietà non appartiene al secondo gruppo
      default:
         return(false);
     }
//--- la proprietà appartiene al secondo gruppo
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Chiamato solo se è si sa già che la proprietà non appartiene |
//| al secondo e terzo gruppo proprietà |
//+------------------------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- la proprietà appartiene al primo gruppo
   group_type=1;
```

```
//--- definisce il tipo di proprietà display
   switch(property_number)
     {
      //--- proprietà integer
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- proprietà double
      case CHART_SHIFT_SIZE:
      case CHART_FIXED_POSITION:
      case CHART_FIXED_MAX:
      case CHART_FIXED_MIN:
      case CHART_POINTS_PER_BAR:
         draw_type='D';
         return;
         //--- solo le proprietà booleane sono rimaste
      default:
         draw_type='B';
         return;
     }
  }
//+------------------------------------------------------------------------------+
//| Crea l'etichetta ed il bottone per la proprietà |
//+------------------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- Array statico per la commutazione all'interno di array di colore ExtColors
   static uint color_index[3]={1,1,1};
//--- indice di cambiamento per la ricezione di un altro colore
   color_index[type]=1-color_index[type];
//--- mostra le etichette ed un bottone (se btn=true) per la proprietà
   if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)ind,ExtNames[ind],ExtColors[colo
      return(false);
   if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)ind,"",ExtColors[color_index[t
      return(false);
   if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
      return(false);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Crea etichette                                                               |
//+------------------------------------------------------------------------------+
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                 const color clr,const int x,const int y)
  {
   if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
```

```
   if(!lbl.Description(text))             return(false);
   if(!lbl.FontSize(10))                  return(false);
   if(!lbl.Color(clr))                    return(false);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Crea bottoni |
//+------------------------------------------------------------------------------------+
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
  {
   if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
   if(!btn.Description("Next"))                  return(false);
   if(!btn.FontSize(10))                         return(false);
   if(!btn.Color(clrBlack))                      return(false);
   if(!btn.BackColor(clrWhite))                  return(false);
   if(!btn.BorderColor(clrBlack))                return(false);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Definisce il massimo ed il minimo valore e step delle proprietà |
//+------------------------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- imposta i valori a seconda del tipo di proprietà
   switch(property_number)
     {
      case CHART_SCALE:
         max=5;
         min=0;
         step=1;
         break;
      case CHART_MODE:
      case CHART_SHOW_VOLUMES:
         max=2;
         min=0;
         step=1;
         break;
      case CHART_SHIFT_SIZE:
         max=50;
         min=10;
         step=2.5;
         break;
      case CHART_FIXED_POSITION:
         max=90;
         min=0;
```

```
         step=15;
         break;
      case CHART_POINTS_PER_BAR:
         max=19;
         min=1;
         step=3;
         break;
      case CHART_FIXED_MAX:
         value=ChartGetDouble(0,CHART_FIXED_MAX);
         max=value*1.25;
         min=value;
         step=value/32;
         break;
      case CHART_FIXED_MIN:
         value=ChartGetDouble(0,CHART_FIXED_MIN);
         max=value;
         min=value*0.75;
         step=value/32;
         break;
      case CHART_HEIGHT_IN_PIXELS:
         max=700;
         min=520;
         step=30;
         break;
         //--- valori default
      default:
         max=1;
         min=0;
         step=1;
     }
  }
```

Costanti Oggetti

Ci sono 44 oggetti grafici che possono essere creati e visualizzati nel chart dei prezzi. Tutte le costanti per lavorare con gli oggetti sono divise in 9 gruppi:

- Tipi di oggetti - Identificatori di oggetti grafici;
- Proprietà dell'oggetto - imposta ed ottiene le proprietà degli oggetti grafici;
- Metodi di associazione oggetto - costanti di posizionamento oggetti nel grafico;
- Angolo binding - indicazione dell'angolo del programma, che è posizionato sull'oggetto;
- Visibilità degli oggetti - indica i timeframes in cui un oggetto è visibile;
- Livelli di Onde di Elliott - marcature di gradazione onda;
- Oggetti Gann - costanti sull'andamento del vantaglio Gann e della griglia Gann;
- Web colors – costanti dei colori web predefiniti;
- Wingdings – codici dei caratteri del font Wingdings.

Tipi di oggetti

Quando un oggetto grafico viene creato utilizzando la funzione ObjectCreate(), è necessario specificare il tipo di oggetto creato, che può essere uno dei valori dell'enumerazione ENUM_OBJECT. Ulteriori specifiche delle proprietà dell'oggetto sono possibili utilizzando le funzioni per lavorare con oggetti grafici.

ENUM_OBJECT

ID		Descrizione
OBJ_VLINE		Linea verticale
OBJ_HLINE		Linea orizzontale
OBJ_TREND		Trend Line
OBJ_TRENDBYANGLE		Trend Line per Angolo
OBJ_CYCLES		Linee Cicliche
OBJ_ARROWED_LINE		Linea con freccia
OBJ_CHANNEL		Canale Equidistante
OBJ_STDDEVCHANNEL		Canale Deviazione Standard
OBJ_REGRESSION		Canale Regressione Lineare
OBJ_PITCHFORK		Andrews' Pitchfork
OBJ_GANNLINE		La Linea Gann
OBJ_GANNFAN		In ventaglio Gann
OBJ_GANNGRID		La griglia Gann
OBJ_FIBO		Ritracciamento di Fibonacci
OBJ_FIBOTIMES		Le Fibonacci Time Zones
OBJ_FIBOFAN		Il ventaglio Fibonacci
OBJ_FIBOARC		L' arco Fibonacci
OBJ_FIBOCHANNEL		Il canale Fibonacci
OBJ_EXPANSION		L'espansione Fibonacci
OBJ_ELLIOTWAVE5		L'onda motiva Elliott
OBJ_ELLIOTWAVE3		L'onda correttiva Elliott
OBJ_RECTANGLE		Rettangolo
OBJ_TRIANGLE		Triangolo
OBJ_ELLIPSE		Ellisse
OBJ_ARROW_THUMB_UP		Pollice su

ID		Descrizione
OBJ_ARROW_THUMB_DOWN		Pollice giu
OBJ_ARROW_UP		Freccia su
OBJ_ARROW_DOWN		Freccia giu
OBJ_ARROW_STOP		Segno di Stop
OBJ_ARROW_CHECK		Segno di Visto
OBJ_ARROW_LEFT_PRICE		Etichetta Prezzo a Sinistra
OBJ_ARROW_RIGHT_PRICE		Etichetta Prezzo a Destra
OBJ_ARROW_BUY		Segno di Buy
OBJ_ARROW_SELL		Segno di Sell
OBJ_ARROW		Freccia
OBJ_TEXT		Testo
OBJ_LABEL		Etichetta
OBJ_BUTTON		Bottone
OBJ_CHART		Grafico
OBJ_BITMAP		Bitmap
OBJ_BITMAP_LABEL		Etichetta Bitmap
OBJ_EDIT		Modifica
OBJ_EVENT		L'oggetto "Evento" corrispondente ad un evento nel calendario economico
OBJ_RECTANGLE_LABEL		L' oggetto "Etichetta Rettangolo" per la creazione e la progettazione dell'interfaccia grafica personalizzata.

OBJ_VLINE

Linea verticale.

Nota

Quando si disegna una linea verticale, è possibile impostare la modalità di visualizzazione della linea per tutte le finestre chart (di proprietà OBJPROP_RAY).

Esempio

Il seguente script crea e sposta la linea verticale sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Linea Verticale\" ."
#property description "La data del punto di ancoraggio è impostata in percentuale dell
#property description "spessoore in barre della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="VLine";     // Nome della linea
input int             InpDate=25;          // Data evento, %
input color           InpColor=clrRed;     // Colore della Linea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Stile della linea
input int             InpWidth=3;          // Spessore della linea
input bool            InpBack=false;       // Sottofondo linea
input bool            InpSelection=true;   // Evdenzia movimento
input bool            InpRay=true;         // Continuazione della linea verso il bass
```

```
input bool            InpHidden=true;      // Nascosto nella lista oggetti
input long            InpZOrder=0;         // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea la linea verticale                                                      |
//+------------------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // ID del chart
                 const string          name="VLine",      // nome della linea
                  const int             sub_window=0,      // indice sottofinestra
                 datetime              time=0,            // orario della linea
                 const color           clr=clrRed,        // colore della linea
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // stile della linea
                 const int             width=1,           // spessore della linea
                 const bool            back=false,        // in sottofondo
                 const bool            selection=true,    // evidenzia movimento
                 const bool            ray=true,          // continuazione della linea
                           const bool            hidden=true,       // nascosto nella
                           const long            z_order=0)         // priorità per il
  {
//--- se l'orario non è impostato, lo disegna per mezzo dell'ultima barra
   if(!time)
      time=TimeCurrent();
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea una linea verticale
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare la linea verticale! Error code = ",GetLastError());
      return(false);
     }
//--- imposta colore della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilitare (true) o disabilitare (false) il modo di spostare la linea con frecci
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) il modo di visualizzazione della linea nella
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta la linea verticale                                                    |
//+------------------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // ID del chart
               const string name="VLine", // nome della linea
               datetime     time=0)       // orario della linea
  {
//--- se l'orario della linea non è impostato, sposta la linea all'ultima barra
   if(!time)
      time=TimeCurrent();
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta la linea verticale
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare la linea verticale! Error code = ",GetLastErr
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina la linea verticale                                                   |
//+------------------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // ID del chart
                 const string name="VLine") // nome della linea
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina la linea verticale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare la linea verticale! Error code = ",GetLastErr
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
```

```
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- array per memorizzare i valori data che devono essere usati
//--- per impostare e cambiare le coordinate del punto di ancoraggio della linea
   datetime date[];
//--- allocazione della memoria
   ArrayResize(date,bars);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- definisce i punti per disegnare la linea
   int d=InpDate*(bars-1)/100;
//--- crea una linea verticale
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta la linea
//---contatore del ciclo
   int h_steps=bars/2;
//--- sposta la linea
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d<bars-1)
         d+=1;
      //--- sposta il punto
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.03 secondi di ritardo
      Sleep(30);
     }
```

```
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il canale dal chart
   VLineDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_HLINE

Linea Orizzontale.

Esempio

Il seguente script crea e sposta la linea orizzontale sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Linea orizzontale\" ."
#property description "Il prezzo del punto di ancoraggio è impostato in percentuale de
#property description "finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="HLine";      // Nome della Linea
input int             InpPrice=25;          // Prezzo della Linea, %
input color           InpColor=clrRed;      // Colore della Linea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Stile della linea
input int             InpWidth=3;           // Spessore della linea
input bool            InpBack=false;        // Sottofondo linea
input bool            InpSelection=true;    // Evdenzia movimento
input bool            InpHidden=true;       // Nascosto nella lista oggetti
input long            InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea la linea orizzontale                                                    |
//+------------------------------------------------------------------------------+
```

```
bool HLineCreate(const long            chart_ID=0,        // ID del chart
                 const string          name="HLine", // nome della linea
                 const int             sub_window=0,      // indice sottofinestra
                 double                price=0,           // prezzo della linea
                 const color           clr=clrRed,        // colore della linea
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // stile della linea
                 const int             width=1,           // spessore della linea
                 const bool            back=false,        // in sottofondo
                 const bool            selection=true,    // evidenzia movimento
                       const bool            hidden=true,       // nascosto nella
                       const long            z_order=0)         // priorità per il
  {
//--- se il prezzo non è impostato, l'imposta all'attuale livello del prezzo Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea una linea orizzontale
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare la linea orizzontale! Error code = ",GetLastError
      return(false);
     }
//--- imposta colore della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilitare (true) o disabilitare (false) il modo di spostare la linea con frecci
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta la linea orizzontale                                                  |
//+------------------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // ID del chart
```

```
                const string name="HLine", // nome della linea
                double        price=0,       // prezzo della linea
  {
//--- se il prezzo non è impostato, lo sposta all'attuale livello di prezzo Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta la linea orizzontale
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare la linea orizzontale! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina la linea orizzontale                                                       |
//+------------------------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // ID del chart
                 const string name="HLine") // line name
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina la linea orizzontale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare la linea orizzontale! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- grandezza dell'array prezzo
   int accuracy=1000;
```

```
//--- array per memorizzare i valori dei prezzi che devono essere usati
//--- per impostare e cambiare le coordinate del punto di ancoraggio della linea
   double price[];
//--- allocazione della memoria
   ArrayResize(price,accuracy);
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare la linea
   int p=InpPrice*(accuracy-1)/100;
//--- crea una linea orizzontale
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta la linea
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- sposta la linea
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p<accuracy-1)
         p+=1;
      //--- sposta il punto
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina dal chart
   HLineDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
```

```
//---
  }
```

OBJ_TREND

Trend Line.

Nota

Per le "Trend Line", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente).

Esempio

Il seguente script crea e sposta la trend linesul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Trend Line\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Trend";     // Nome della linea
input int             InpDate1=35;         // Data del 1mo punto, in %
input int             InpPrice1=60;        // Prezzo del 1mo punto, in%
input int             InpDate2=65;         // data del 2ndo punto, %
input int             InpPrice2=40;        // Prezzo del 2ndo punto, in %
input color           InpColor=clrRed;     // Colore della Linea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Stile della linea
```

```
input int              InpWidth=2;          // Spessore linea
input bool             InpBack=false;       // Sottofondo linea
input bool             InpSelection=true;   // Evdenzia movimento
input bool             InpRayLeft=false;    // Continuazione della linea a sinistra
input bool             InpRayRight=false;   // Continuazione della linea a destra
input bool             InpHidden=true;      // Nascosto nella lista oggetti
input long             InpZOrder=0;         // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea la trendline dalle coordinate fornite                                   |
//+------------------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // ID del chart
                 const string          name="TrendLine",  // nome della linea
                 const int             sub_window=0,      // indice sottofinestra
                 datetime              time1=0,           // orario del primo punto
                 double                price1=0,          // prezzo del primo punto
                 datetime              time2=0,           // orario del secondo punto
                 double                price2=0,          // prezzo del secondo punto
                 const color           clr=clrRed,        // colore della linea
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // stile della linea
                 const int             width=1,           // spessore della linea
                 const bool            back=false,        // in sottofondo
                 const bool            selection=true,    // evidenzia movimento
                 const bool            ray_left=false,    // continuazione della linea
                 const bool            ray_right=false,   // continuazione della linea
                 const bool            hidden=true,       // nascosto nella lista ogge
                 const long            z_order=0)         // priorità per il click del
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea una trend line delle coordinate fornite
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare una trend line! Error code = ",GetLastError());
      return(false);
     }
//--- imposta colore della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilitare (true) o disabilitare (false) il modo di spostare la linea con frecci
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
```

```
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio della trend line                               |
//+------------------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID del chart
                      const string name="TrendLine", // nome dellal linea
                      const int    point_index=0,    // indice del punto di ancoraggio
                  datetime     time=0,          // coordinate tempo, del punto di anco
                   double       price=0)          // coordinate prezzo, del punto di a
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio della trendline
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| La funzione rimuove la trendline dal chart.                                  |
//+------------------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID del chart
                 const string name="TrendLine") // nome della linea
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina una trendline
```

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare una trendline! Error code = ",GetLastError())
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio della trendline ed imposta i          |
//| valori di default per quelli vuoti                                                 |
//+------------------------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, è posizionato 9 barre meno dalla
   if(!time2)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- imposta il secondo punto, 9 barre in meno dalla prima
      time2=temp[0];
     }
//--- se il prezzo del secondo punto non è impostato, è uguale a quello del primo punt
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

```
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio della linea
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare la linea
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea una trend line
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio della linea
//---contatore del ciclo
   int v_steps=accuracy/5;
//--- sposta il primo punto di ancoraggio, verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1>1)
         p1-=1;
      //--- sposta il punto
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
```

```
         return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
         return;
       //--- ridisegna il chart
       ChartRedraw();
      }
//--- sposta il secondo punto di ancoraggio, verticalmente
   for(int i=0;i<v_steps;i++)
     {
       //--- usa il seguente valore
       if(p2<accuracy-1)
         p2+=1;
       //--- sposta il punto
       if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
         return;
       //--- ridisegna il chart
       ChartRedraw();
      }
//--- ritardo di mezzo secondo
   Sleep(500);
//---contatore del ciclo
   int h_steps=bars/2;
//--- sposta entrambi i punti di ancoraggio, orizzontalmente allo stesso orario
   for(int i=0;i<h_steps;i++)
     {
       //--- usa i seguenti valori
       if(d1<bars-1)
         d1+=1;
       if(d2>1)
         d2-=1;
       //--- slitta i punti
       if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
       if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
         return;
       //--- ridisegna il chart
       ChartRedraw();
       // 0.03 secondi di ritardo
       Sleep(30);
      }
//--- 1 secondo di ritardo
   Sleep(1000);
```

```
//--- elimina una trendline
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_TRENDBYANGLE

Trendline per Angolo.

Nota

Per le "Trend Line per Angolo", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente).

Sia angolo e le coordinate del secondo punto di ancoraggio possono essere utilizzate per impostare la pendenza della linea.

Esempio

Il seguente script crea e sposta la trend linesul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Trend Line Angolata\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description "finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Trend";      // Nome della linea
input int             InpDate1=50;          // data del 1mo punto, %
input int             InpPrice1=75;         // prezzo del 1mo punto, %
input int             InpAngle=0;           // Angolo di pendenza della linea
input color           InpColor=clrRed;      // Colore della Linea
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Stile della linea
input int             InpWidth=2;          // Spessore linea
input bool            InpBack=false;       // Sottofondo linea
input bool            InpSelection=true;   // Evdenzia movimento
input bool            InpRayLeft=false;    // Continuazione della linea a sinistra
input bool            InpRayRight=true;    // Continuazione della linea a destra
input bool            InpHidden=true;      // Nascosto nella lista oggetti
input long            InpZOrder=0;         // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea una trendline angolata                                                  |
//+------------------------------------------------------------------------------+
bool TrendByAngleCreate(const long            chart_ID=0,        // ID del chart
                        const string          name="TrendLine",  // nome della linea
                        const int             sub_window=0,      // indice sottofinest
                        datetime              time=0,            // orario del punto
                        double                price=0,           // prezzo del punto
                        const double          angle=45.0,        // angolo di pendenza
                        const color           clr=clrRed,        // colore della linea
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // stile della linea
                        const int             width=1,           // spessore della lin
                        const bool            back=false,        // in sottofondo
                        const bool            selection=true,    // evidenzia moviment
                        const bool            ray_left=false,    // continuazione dell
                        const bool            ray_right=false,   // continuazione dell
                        const bool            hidden=true,       // nascosto nella lis
                        const long            z_order=0)         // priorità per il cl
  {
//-- crea il secondo punto per facilitare il trascinamento della trendline con il mous
   datetime time2=0;
   double   price2=0;
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea una trendline usando 2 punti
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2)
     {
      Print(__FUNCTION__,
            ": fallimento nel creare una trend line! Error code = ",GetLastError());
      return(false);
     }
//--- cambia l'angolo di pendenza della trendline; quando si cambia l'angolo, le coord
//--- punto delle linea vengono ridefinite automaticamente secondo i nuovi valori del
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- imposta colore della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilitare (true) o disabilitare (false) il modo di spostare la linea con frecci
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia i punti di ancoraggio delle coordinate delle trendline                |
//+------------------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID del chart
                      const string name="TrendLine", // nome dellal linea
                      datetime     time=0,           // coordinate tempo, del punto di
                      double       price=0)          // coordinate prezzo, del punto d
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio della trendline
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia l'angolo di pendenza della trendline                                  |
//+------------------------------------------------------------------------------+
bool TrendAngleChange(const long   chart_ID=0,       // ID del chart
```

```
                      const string name="TrendLine", // nome della trend line
                      const double angle=45)         // angolo di pendenza della trend
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia l'angolo di pendenza della trendline
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'angolo di pendenza della linea! Error code =
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina la trendline                                                         |
//+------------------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID del chart
                 const string name="TrendLine") // nome della linea
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina una trendline
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare una trendline! Error code = ",GetLastError())
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio della trendline ed imposta i      |
//| valori di default per quelli vuoti                                           |
//+------------------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- imposta le coordinate del secondo punto ausiliare
//--- il secondo punto sarà 9 barre a sinistra ed avrà lo stesso prezzo
   datetime second_point_time[10];
```

```
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio della linea
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare la linea
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crea una trend line
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
```

```
      }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta e ruota la linea
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- sposta il punto di ancoraggio e cambia l'angolo di pendenza della linea
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1>1)
         p1-=1;
      //--- sposta il punto
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina dal chart
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_CYCLES

Linee Ciclo.

Nota

La distanza tra le linee è impostata da coordinate temporali dei due punti di ancoraggio dell'oggetto.

Esempio

Il seguente script crea e sposta le linee ciclo sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script crea le linee ciclo sul chart."
#property description "Coordinate dei punti di ancoraggio sono impostare in percentual
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Cycles";   // nome dell'oggetto
input int             InpDate1=10;        // data del 1mo punto, %
input int             InpPrice1=45;       // prezzo del 1mo punto, %
input int             InpDate2=20;        // data del 2ndo punto, %
input int             InpPrice2=55;       // prezzo del 2ndo punto, %
input color           InpColor=clrRed;    // Colore delle linee ciclo
input ENUM_LINE_STYLE InpStyle=STYLE_DOT; // Stile delle linee ciclo
input int             InpWidth=1;         // Spessore delle linee ciclo
```

```
input bool            InpBack=false;      // Oggetto di sottofondo
input bool            InpSelection=true;  // Evidenzia movimento
input bool            InpHidden=true;     // Nascosto nella lista oggetti
input long            InpZOrder=0;        // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea linee ciclo                                                             |
//+------------------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,        // ID del chart
                  const string          name="Cycles",     // nome dell'oggetto
                  const int             sub_window=0,      // indice sottofinestra
                  datetime              time1=0,           // orario del primo punto
                  double                price1=0,          // prezzo del primo punto
                  datetime              time2=0,           // orario del secondo punto
                  double                price2=0,          // prezzo del secondo punto
                  const color           clr=clrRed,        // colore delle linee ciclo
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee ciclo
                  const int             width=1,           // spessore delle linee ci
                  const bool            back=false,        // in sottofondo
                  const bool            selection=true,    // evidenzia movimento
                  const bool            hidden=true,       // nascosto nella lista og
                  const long            z_order=0)         // priorità per il click de
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea le linee ciclo dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare le linee ciclo! Error code = ",GetLastError());
      return(false);
     }
//--- imposta i colori delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del display delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostamento delle linee per m
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

```
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,    // ID del chart
                       const string name="Cycles", // nome dell'oggetto
                       const int    point_index=0, // indice del punto di ancoraggio
                       datetime     time=0,        // coordinate del punto di ancorag
                       double       price=0)       // coordinate del punto di ancorag
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina le linee ciclo                                                       |
//+------------------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,    // ID del chart
                  const string name="Cycles") // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina le linee ciclo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare le linee ciclo! Error code = ",GetLastError()
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

```
//+------------------------------------------------------------------------------+
//| Imposta i valori dei punti di ancoraggio delle linee ciclo ed imposta i      |
//| valori di default per quelli vuoti                                           |
//+------------------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, è posizionato 9 barre meno dalla
   if(!time2)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- imposta il secondo punto, 9 barre in meno dalla prima
      time2=temp[0];
     }
//--- se il prezzo del secondo punto non è impostato, è uguale a quello del primo punt
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio delle linee ciclo
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

```
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per il disegno delle linee ciclo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea una trend line
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio
//---contatore del ciclo
   int h_steps=bars/5;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d2<bars-1)
         d2+=1;
      //--- sposta il punto
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
```

```
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   h_steps=bars/4;
//--- move the first anchor point
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d1<bars-1)
         d1+=1;
      //--- sposta il punto
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'oggetto dal chart
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROWED_LINE

Linea con freccia

Esempio

Il seguente script crea e sposta una linea con freccia sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Linea con freccia\" ."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="ArrowedLine"; // nome della Linea
input int             InpDate1=35;           // Data del 1mo punto, in %
input int             InpPrice1=60;          // Prezzo del 1mo punto, in%
input int             InpDate2=65;           // Data del 2ndo punto, in %
input int             InpPrice2=40;          // Prezzo del 2ndo punto, in %
input color           InpColor=clrRed;       // Colore della linea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // Stile della linea
input int             InpWidth=2;            // Spessore della linea
input bool            InpBack=false;         // Linea di sottofondo
input bool            InpSelection=true;     // Evidenzia spostamento
input bool            InpHidden=true;        // Nascosto nella lista oggetti
input long            InpZOrder=0;           // Priorità per il click del mouse
```

```
//+------------------------------------------------------------------------------+
//| Crea un linea con freccia, dalle coordinate date                             |
//+------------------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,         // ID del chart
                       const string          name="ArrowedLine", // nome della linea
                       const int             sub_window=0,       // indice sottofinest
                       datetime              time1=0,            // orario del primo p
                       double                price1=0,           // prezzo del primo p
                       datetime              time2=0,            // orario del secondo
                       double                price2=0,           // prezzo del secondo
                       const color           clr=clrRed,         // colore della linea
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // stile della linea
                       const int             width=1,            // spessore della lin
                       const bool            back=false,         // in sottofondo
                       const bool            selection=true,     // evidenzia moviment
                       const bool            hidden=true,        // nascosto nella lis
                       const long            z_order=0)          // priorità per il cl
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea una linea con freccia dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            ": fallimento nel creare la linea con freccia! Error code = ",GetLastError
      return(false);
     }
//--- imposta colore della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilitare (true) o disabilitare (false) il modo di spostare la linea con freccia
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
```

```
  }
//+------------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio della linea con freccia                              |
//+------------------------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,          // ID del chart
                            const string name="ArrowedLine", // nome della linea
                            const int    point_index=0,      // indice del punto di ar
                            datetime     time=0,             // coordinate tempo, del
                            double       price=0)            // coordinate prezzo, del
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio della linea
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| La funzione rimuove la linea con freccia dal chart                                 |
//+------------------------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,          // ID del chart
                       const string name="ArrowedLine") // nome della linea
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina una linea con freccia
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare la linea con freccia! Error code = ",GetLastError
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori del punto di ancoraggio ed imposta i valori di default          |
//| per quelli vuoti                                                                   |
//+------------------------------------------------------------------------------------+
```

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, è posizionato 9 barre meno dalla
   if(!time2)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- imposta il secondo punto, 9 barre in meno dalla prima
      time2=temp[0];
     }
//--- se il prezzo del secondo punto non è impostato, è uguale a quello del primo punt
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio della linea
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

```
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare la linea
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea la linea con freccia
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio della linea
//---contatore del ciclo
   int v_steps=accuracy/5;
//--- sposta il secondo punto di ancoraggio, verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p2<accuracy-1)
         p2+=1;
      //--- sposta il punto
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- sposta il primo punto di ancoraggio, verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1>1)
         p1-=1;
```

```
      //--- sposta il punto
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//---contatore del ciclo
   int h_steps=bars/2;
//--- sposta entrambi i punti di ancoraggio, orizzontalmente allo stesso orario
   for(int i=0;i<h_steps;i++)
     {
      //--- usa i seguenti valori
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- slitta i punti
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.03 secondi di ritardo
      Sleep(30);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina una linea con freccia
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_CHANNEL

Canale Equidistante

Nota

Per un canale equidistante, è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente). Può essere anche impostata la modalità di riempimento del canale con colore.

Esempio

Il seguente script crea e muove un canale equidistante sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Script draws \"Equidistant Channel\" oggetto grafico."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description "finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Channel";   // Nome del canale
input int             InpDate1=25;         // data del 1mo punto, %
input int             InpPrice1=60;        // Prezzo del 1mo punto, in%
input int             InpDate2=65;         // data del 2ndo punto, %
input int             InpPrice2=80;        // prezzo del 2ndo punto, %
input int             InpDate3=30;         // data del 3zo punto, %
```

```
input int              InpPrice3=40;         // prezzo del 3zo punto, %
input color            InpColor=clrRed;      // Colore del canale
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Stile delle linee del canale
input int              InpWidth=2;           // Spessore della linea del canale
input bool             InpBack=false;        // Canale di sottofondo
input bool             InpFill=false;        // Riempie il canale con colore
input bool             InpSelection=true;    // Evdenzia movimento
input bool             InpRayLeft=false;     // Continuazione del canale a sinistra
input bool             InpRayRight=false;    // Continuazione del canale a destra
input bool             InpHidden=true;       // Nascosto nella lista oggetti
input long             InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea un canale equidistante dalle coordinate date                            |
//+------------------------------------------------------------------------------+
bool ChannelCreate(const long            chart_ID=0,        // ID del chart
                   const string          name="Channel",    // nome del canale
                   const int             sub_window=0,      // indice sottofinestra
                   datetime              time1=0,           // orario del primo punto
                   double                price1=0,          // prezzo del primo punto
                   datetime              time2=0,           // orario del secondo punt
                   double                price2=0,          // prezzo del secondo punt
                   datetime              time3=0,           // orario del terzo punto
                   double                price3=0,          // prezzo del terzo punto
                   const color           clr=clrRed,        // colore del canale
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee del
                   const int             width=1,           // spesso delle linee del
                   const bool            fill=false,        // riempimento del canale
                   const bool            back=false,        // in sottofondo
                   const bool            selection=true,    // evidenzia movimento
                   const bool            ray_left=false,    // continua del canale a
                   const bool            ray_right=false,   // continua del canale a
                   const bool            hidden=true,       // nascosto nella lista o
                   const long            z_order=0)         // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il canale dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il canale equidistante! Error code = ",GetLastErr
      return(false);
     }
//--- imposta il colore del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

```
//--- imposta lo spessore delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- abilita (true) o disabilita (false) la modalità di riempimento del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio del canale                                           |
//+------------------------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,      // ID del chart
                        const string name="Channel", // nome del canale
                        const int    point_index=0,  // indice del punto di ancoraggio
                        datetime     time=0,         // coordinate del punto di ancora
                        double       price=0)        // coordinate prezzo, del punto d
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
```

```
//+------------------------------------------------------------------------------------+
//| Elimina il canale                                                                  |
//+------------------------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,     // ID del chart
                   const string name="Channel") // nome del canale
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il canale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il canale! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio del canale ed impostare i valori di dei
//| per quelli vuoti
//+------------------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- se l'orario del secondo(destro) punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se il prezzo del secondo punto non è impostato, avrà un valore Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del primo(sinistro) punto non è impostato, è posizionato 9 barre mer
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del primo punto non è impostato, lo sposta di 300 punti in più risp
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se l'orario del terzo punto non è impostato, esso coincide con quello del primo
   if(!time3)
      time3=time1;
//--- se il prezzo del terzo punto non è impostato, è uguale a quello del secondo punt
   if(!price3)
      price3=price2;
```

```
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e modificare le coordinate dei punti di ancoraggio del canale"
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il canale
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crea il canale equidistante
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
```

```
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio del canale
//---contatore del ciclo
   int h_steps=bars/6;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d2<bars-1)
         d2+=1;
      //--- sposta il punto
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- move the first anchor point
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d1>1)
         d1-=1;
      //--- sposta il punto
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
```

```
   int v_steps=accuracy/10;
//--- sposta il terzo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p3>1)
         p3-=1;
      //--- sposta il punto
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il canale dal chart
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_STDDEVCHANNEL

Canale di Deviazione Strandard.

Nota

Per il "Canale di Deviazione Strandard", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente). Può essere anche impostata la modalità di riempimento del canale con colore.

OBJPROP_DEVIATION la proprietà è utilizzata per cambiare il valore della deviazione del canale.

Esempio

Il seguente script crea e sposta un Canale di Regressione Lineare sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Canale di Deviazione Star
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description "finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="StdDevChannel";    // Nome del Canale
input int             InpDate1=10;                // data del 1mo punto, %
input int             InpDate2=40;                // data del 2ndo punto, %
input double          InpScale=1.0;               // Deviazione
input color           InpColor=clrRed;            // Colore del canale
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile delle linee del canale
input int             InpWidth=2;                // Spessore delle linee del canale
input bool            InpFill=false;             // Riempie il canale con colore
input bool            InpBack=false;             // Canale di sottofondo
input bool            InpSelection=true;         // Evidenzia movimento
input bool            InpRayLeft=false;          // Continuazione del canale a sinistr
input bool            InpRayRight=false;         // Continuazione del canale a destra
input bool            InpHidden=true;            // Nascosto nella lista oggetti
input long            InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il canale di deviazione standard dalle coordinate fornite |
//+------------------------------------------------------------------------------+
bool StdDevChannelCreate(const long            chart_ID=0,        // ID del chart
                         const string          name="Channel",    // nome del canale
                         const int             sub_window=0,      // indice sottofines
                         datetime              time1=0,           // orario del primo
                         datetime              time2=0,           // orario del secon
                         const double          deviation=1.0,     // deviazione
                         const color           clr=clrRed,        // colore del canale
                         const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee
                         const int             width=1,           // spessore delle li
                         const bool            fill=false,        // riempie il canale
                         const bool            back=false,        // in sottofondo
                         const bool            selection=true,    // evidenzia movimen
                         const bool            ray_left=false,    // continuazione del
                         const bool            ray_right=false,   // continuazione del
                         const bool            hidden=true,       // nascosto nella li
                         const long            z_order=0)         // priorità per il c
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeChannelEmptyPoints(time1,time2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il canale dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il canale di deviazione strandard! Error code = "
      return(false);
     }
//--- imposta i valori di deviazione che interessano lo spessore del canale
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- imposta il colore del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- abilita (true) o disabilita (false) la modalità di riempimento del canale
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio del canale                                     |
//+------------------------------------------------------------------------------+
bool StdDevChannelPointChange(const long   chart_ID=0,      // ID del chart
                              const string name="Channel", // nome del canale
                              const int    point_index=0,  // indice del punto di anco
                              datetime     time=0,         // coordinate tempo, del pu
  {
//--- se l'orario non è impostato, sposta il punto sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia la deviazione del canale                                              |
//+------------------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,      // ID del chart
                                  const string name="Channel", // nome del canale
                                  const double deviation=1.0)  // deviation
```

```
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia l'angolo di pendenza della trendline
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare la deviazione del canale! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il canale                                                                  |
//+------------------------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,     // ID del chart
                         const string name="Channel") // nome del canale
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il canale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il canale! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio del canale ed impostare i valori di de
//| per quelli vuoti
//+------------------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
```

```
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e modificare le coordinate dei punti di ancoraggio del canale"
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il canale
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- crea la deviazione standard del canale
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpSty
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
```

```
   ChartRedraw();
   Sleep(1000);
//--- ora, il canale orizzontalmente sulla destra ed espanso
//---contatore del ciclo
   int h_steps=bars/2;
//--- sposta il canale
   for(int i=0;i<h_steps;i++)
     {
      //--- usa i seguenti valori
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- move the anchor points
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   double v_steps=InpDeviation*2;
//--- espande il canale
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il canale dal chart
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
```

```
}
```

OBJ_REGRESSION

Canale di Regressione Lineare.

Nota

Per il "Canale di Regressione Lineare", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente). Può essere anche impostata la modalità di riempimento del canale con colore.

Esempio

Il seguente script crea e muove un canale di regressione lineare sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Canale di Regressione Lir
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description "finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Regression"; // Nome del canale
input int             InpDate1=10;          // data del 1mo punto, %
input int             InpDate2=40;          // data del 2ndo punto, %
input color           InpColor=clrRed;      // Colore del canale
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Stile delle linee del canale
input int             InpWidth=2;           // Spessore delle linee del canale
```

```
input bool             InpFill=false;        // Riempie il canale con colore
input bool             InpBack=false;        // Background channel
input bool             InpSelection=true;    // evidenza spostamento
input bool             InpRayLeft=false;     // Continuazione del canale a sinistra
input bool             InpRayRight=false;    // Continuazione del canale a destra
input bool             InpHidden=true;       // Nascosto nella lista oggetti
input long             InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il Canale di Regressione Lineare dalle coordinate fornite |
//+------------------------------------------------------------------------------+
bool RegressionCreate(const long             chart_ID=0,        // ID del chart
                      const string           name="Regression", // nome del canale
                      const int              sub_window=0,      // indice della sottof
                      datetime               time1=0,           // orario del primo pun
                      datetime               time2=0,           // orario del secondo p
                      const color            clr=clrRed,        // colore del canale
                      const ENUM_LINE_STYLE  style=STYLE_SOLID, // stile delle linee de
                      const int              width=1,           // spessore delle linee
                      const bool             fill=false,        // riempimento del cana
                      const bool             back=false,        // in sottofondo
                      const bool             selection=true,    // evidenza movimento
                      const bool             ray_left=false,    // continua del canale
                      const bool             ray_right=false,   // continua del canale
                      const bool             hidden=true,       // nascosto nella lista
                      const long             z_order=0)         // priorità per il clic
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeRegressionEmptyPoints(time1,time2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il canale dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il canale di regressione lineare! Error code = ",
      return(false);
     }
//--- imposta il colore del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- abilita (true) o disabilita (false) la modalità di riempimento del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
```

```
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio del canale                                     |
//+------------------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,     // ID del chart
                           const string name="Channel", // nome del canale
                           const int    point_index=0,  // indice del punto di ancorag
                           datetime     time=0,         // coordinate tempo, del punto
  {
//--- se l'orario non è impostato, sposta il punto sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina il canale                                                            |
//+------------------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,     // ID del chart
                      const string name="Channel") // nome del canale
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il canale
   if(!ObjectDelete(chart_ID,name))
     {
```

```
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il canale! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Controlla i valori dei punti di ancoraggio del canale ed impostare i valori di dei
//| per quelli vuoti
//+------------------------------------------------------------------------------------
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e modificare le coordinate dei punti di ancoraggio del canale"
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

```
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il canale
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- crea un canale di regressione lineare
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il canale orizzontalmente sulla destra
//---contatore del ciclo
   int h_steps=bars/2;
//--- sposta il canale
   for(int i=0;i<h_steps;i++)
     {
      //--- usa i seguenti valori
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- move the anchor points
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
```

```
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il canale dal chart
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_PITCHFORK

Pitchfork di Andrew.

Nota

Per le Andrews' Pitchfork, è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente).

È inoltre possibile specificare il numero di linee-livelli, i loro valori ed il colore.

Esempio

Il seguente script crea e sposta le Andrews' Pitchfork sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \" Andrews' Pitchfork\" ."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Pitchfork";  // Nome della Pitchfork
input int             InpDate1=14;          // data del 1mo punto, %
input int             InpPrice1=40;         // prezzo del 1mo punto, %
input int             InpDate2=18;          // data del 2ndo punto, %
input int             InpPrice2=50;         // prezzo del 2ndo punto, %
input int             InpDate3=18;          // data del 3zo punto, %
input int             InpPrice3=30;         // prezzo del 3zo punto, %
```

```
input color           InpColor=clrRed;      // Colore della Pitchfork
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Stile delle linee pitchfork
input int             InpWidth=1;           // Spessore delle linee pitchfork
input bool            InpBack=false;        // Sottofondo delle pitchfork
input bool            InpSelection=true;    // evidenza spostamento
input bool            InpRayLeft=false;     // Continua delle Pitchfork's sulla sinist
input bool            InpRayRight=false;    // Continua delle Pitchfork's sulla destra
input bool            InpHidden=true;       // Nascosto nella lista oggetti
input long            InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea le Andrews' Pitchfork dalle coordinate date                             |
//+------------------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // ID del chart
                     const string          name="Pitchfork",  // nome della pitchfork
                     const int             sub_window=0,      // indice sottofinestra
                     datetime              time1=0,           // orario del primo punt
                     double                price1=0,          // prezzo del primo punt
                     datetime              time2=0,           // orario del secondo pu
                     double                price2=0,          // prezzo del secondo pu
                     datetime              time3=0,           // orario del terzo punt
                     double                price3=0,          // prezzo del terzo punt
                     const color           clr=clrRed,        // colore delle linee de
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee del
                     const int             width=1,           // spessore delle linee
                     const bool            back=false,        // in sottofondo
                     const bool            selection=true,    // evidenza movimento
                     const bool            ray_left=false,    // continua delle pitchf
                     const bool            ray_right=false,   // continua delle pitchf
                     const bool            hidden=true,       // nascosto nella lista
                     const long            z_order=0)         // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea le Andrews' Pitchfork dalle coordinate fornite
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": fallimento nel creare le \"Andrews' Pitchfork\"! Error code = ",GetLast
      return(false);
     }
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia delle pitchfork pe
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'ogget
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione de
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualiz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualiz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli o
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il numero di livelli delle Andrews' Pitchfork ed i loro parametri    |
//+------------------------------------------------------------------------------+
bool PitchforkLevelsSet(int             levels,           // numero di linee di livel
                        double          &values[],        // valore delle lin
                  color           &colors[],        // colore delle linee livello
                   ENUM_LINE_STYLE &styles[],        // stile delle linee livello
                        int             &widths[],        // spessore delle l
                        const long      chart_ID=0,       // ID del chart
                     const string    name="Pitchfork") // nome della pitchfork
  {
//--- verifica la grandezza dell'array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la lungezza dell'array non corrisponde al numero di livel
      return(false);
     }
//--- set the number of levels
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- imposta le proprietà dei livelli nel loop
   for(int i=0;i<levels;i++)
     {
      //--- valore dei livelli
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- colore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- stile dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- spessore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
```

```
      //--- descrizione dei livelli
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio Andrews' Pitchfork                             |
//+------------------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // ID del chart
                          const string name="Pitchfork", // nome del canale
                          const int    point_index=0,    // indice del punto di ancora
                          datetime     time=0,           // coordinate orarie del punt
                          double       price=0)          // coordinate di prezzo del p
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina la Andrews' Pitchfork                                                |
//+------------------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // ID del chart
                     const string name="Pitchfork") // nome del canale
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il canale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare la \"Andrews' Pitchfork\"! Error code = ",Get
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
```

```
  }
//+------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio della Andrews' Pitchfork ed imposta|
//| i valori di default per quelli vuoti                                          |
//+------------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- se l'orario del secondo punto (superiore destro) non è impostato, sarà sulla ba
   if(!time2)
      time2=TimeCurrent();
//--- se il prezzo del secondo punto non è impostato, avrà un valore Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del primo(sinistro) punto non è impostato, è posizionato 9 barre me
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del primo punto non è impostato, lo sposta di 200 punti sotto al s
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se l'orario del terzo punto non è impostato, esso coincide con quello del secon
   if(!time3)
      time3=time2;
//--- se il prezzo del terzo punto non è impostato, lo sposta di 300 punti in meno ri
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                        |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
```

```
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio delle Andrews' P
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare le Andrews' Pitchfork
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crea la pitchfork
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio della pitchfork
//---contatore del ciclo
   int v_steps=accuracy/10;
//--- move the first anchor point
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1>1)
         p1-=1;
      //--- sposta il punto
```

```
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int h_steps=bars/8;
//--- sposta il terzo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d3<bars-1)
         d3+=1;
      //--- sposta il punto
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy/10;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p2>1)
         p2-=1;
      //--- sposta il punto
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
```

```
      }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina la pitchfork dal chart
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_GANNLINE

Linea di Gann.

Nota

Per la "Linea di Gann", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente).

Entrambi gli ancogli di Gann con una scala e coordinate del secondo punto di ancoraggio possono essere usate per impostare la pendenza della linea.

Esempio

Il seguente script crea e sposta la griglia di Gann sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Linea di Gann\" ."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="GannLine";        // Nome della Linea
input int             InpDate1=20;               // data del 1mo punto, %
input int             InpPrice1=75;              // prezzo del 1mo punto, %
input int             InpDate2=80;               // data del 2ndo punto, %
input double          InpAngle=0.0;              // Angolo di Gann
```

```
input double           InpScale=1.0;              // Scala
input color            InpColor=clrRed;           // Colore della linea
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Stile della linea
input int              InpWidth=2;                // Spessore della linea
input bool             InpBack=false;             // Linea di sottofondo
input bool             InpSelection=true;         // Evidenzia movimento
input bool             InpRayLeft=false;          // Continuazione del canale a sinistr
input bool             InpRayRight=true;          // Continuazione del canale a destra
input bool             InpHidden=true;            // Nascosto nella lista oggetti
input long             InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------------+
//| Crea la Linea di Gann dalle coordinate fornite |
//+------------------------------------------------------------------------------------+
bool GannLineCreate(const long             chart_ID=0,        // ID del chart
                    const string           name="GannLine",   // nome della linea
                    const int              sub_window=0,      // indice sottofinestra
                    datetime               time1=0,           // orario del primo punto
                    double                 price1=0,          // prezzo del primo punto
                    datetime               time2=0,           // orario del secondo pun
                    const double           angle=1.0,         // Angolo di Gann
                    const double           scale=1.0,         // scala
                    const color            clr=clrRed,        // Colore della linea
                    const ENUM_LINE_STYLE  style=STYLE_SOLID, // stile della linea
                    const int              width=1,           // spessore della linea
                    const bool             back=false,        // in sottofondo
                    const bool             selection=true,    // evidenzia movimento
                    const bool             ray_left=false,    // continuazione della li
                    const bool             ray_right=false,   // continuazione della li
                    const bool             hidden=true,       // nascosta nella lista d
                    const long             z_order=0)         // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea la Linea di Gann dalle coordinate date
//--- le giuste coordinate del secondo punto di ancoraggio vengono ridefinite
//--- automaticamente dopo l'angolo di Gann e/o i cambiamenti della scala,
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare la \"Linea di Gann\"! Error code = ",GetLastError
      return(false);
     }
//--- cambia l'angolo di Gann
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- cambia la scala (numero di pips per barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- imposta colore della linea
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenziazione delle linee pe
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli oc
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio della Linea di Gann                             |
//+------------------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,       // ID del chart
                         const string name="GannLine", // nome della linea
                         const int    point_index=0, // indice del punto di ancoraggio
                   datetime     time=0,          // coordinate tempo, del punto di anco
                          double       price=0)           // coordinate di prezzo del p
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio della linea
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
```

```
  }
//+------------------------------------------------------------------------------------+
//| Cambia l'angolo di Gann                                                            |
//+------------------------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,      // ID del chart
                         const string name="GannLine", // nome della linea
                           const double angle=1.0)       // Angolo di Gann
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia l'angolo di Gann
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'angolo di Gann! Error code = ",GetLastError()
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia la scala della Linea di Gann                                                |
//+------------------------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,      // ID del chart
                         const string name="GannLine", // nome della linea
                           const double scale=1.0)       // scala
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la scala (numero di pips per barra)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare la scala! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| La funzione rimuove La Linea di Gann dal chart                                     |
//+------------------------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,      // ID del chart
                    const string name="GannLine") // nome della linea
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina la linea di Gann
   if(!ObjectDelete(chart_ID,name))
```

```
      {
       Print(__FUNCTION__,
             ": fallimento nell'eliminare la \"Linea di Gann\"! Error code = ",GetLastE
       return(false);
      }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio della Griglia di Gann ed imposta i  |
//| valori di default per quelli vuoti                                             |
//+------------------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                         |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio della linea
   datetime date[];
```

```
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare la Linea di Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crea la Linea di Gann
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio della linea e cambia l'angolo
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- sposta il primo punto di ancoraggio, verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1>1)
         p1-=1;
      //--- sposta il punto
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
```

```
      }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- definisce il corrente valore dell'angolo di Gann (cambiato
//--- dopo aver spostato il primo punto di ancoraggio)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//---contatore del ciclo
   v_steps=accuracy/8;
//--- cambia l'angolo di Gann
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina la linea dal chart
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_GANNFAN

Ventagli di Gann.

Nota

Per i Ventagli di Gann, è possibile specificare il tipo di trend dall'enumerazione ENUM_GANN_DIRECTION. Regolando il valore di scala (OBJPROP_SCALE), è possibile cambiare l'angolo di inclinazione delle linee ventaglio.

Esempio

Il seguente script crea e sposta il Ventaglio di Gann sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Ventaglio di Gann\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="GannFan";         // Nome dell'oggetto
input int             InpDate1=15;               // data del 1mo punto, %
input int             InpPrice1=25;              // prezzo del 1mo punto, %
input int             InpDate2=85;               // data del 2ndo punto, %
input double          InpScale=2.0;              // Scala
input bool            InpDirection=false;        // Direzione del Trend
```

```
input color           InpColor=clrRed;           // Colore del Ventaglio
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile delle linee ventaglio
input int             InpWidth=1;                // Spessore delle linee ventaglio
input bool            InpBack=false;             // ventagli di sottofondo
input bool            InpSelection=true;         // Evidenzia movimento
input bool            InpHidden=true;            // Nascosto nella lista oggetti
input long            InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea Ventaglio di Gann |
//+------------------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // ID del chart
                   const string          name="GannFan",    // nome del ventaglio
                   const int             sub_window=0,      // indice sottofinestra
                   datetime              time1=0,           // orario del primo punto
                   double                price1=0,          // prezzo del primo punto
                   datetime              time2=0,           // orario del secondo punt
                   const double          scale=1.0,         // scala
                   const bool            direction=true,    // direzione del trend
                   const color           clr=clrRed,        // colore ventaglio
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee venta
                   const int             width=1,           // spessore delle linee ve
                   const bool            back=false,        // in sottofondo
                   const bool            selection=true,    // evidenzia movimento
                   const bool            hidden=true,       // nascosto nella lista og
                   const long            z_order=0)         // priorità per il click d
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- create Gann Fan by the given coordinates
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il \"Ventaglio di Gann\"! Error code = ",GetLastE
      return(false);
     }
//--- cambia la scala (numero di pips per barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- cambia la direzione del trend del Ventaglio di Gann (true - discendente, false -
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- imposta il colore del ventaglio
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della visualizzazione delle linee del ventaglio
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee del ventaglio
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

```
//--- abilita (true) o disabilita (false) la modalità di evidenzia del venglio per lo
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio del Ventaglio di Gann                          |
//+------------------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,     // ID del chart
                        const string name="GannFan", // nome del ventaglio
                        const int    point_index=0,  // indice del punto di ancoraggio
                        datetime     time=0,         // coordinate del punto di ancora
                        double       price=0)        // coordinate prezzo, del punto d
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio del ventaglio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia la scala del Ventaglio di Gann                                        |
//+------------------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,     // ID del chart
                        const string name="GannFan", // nome del ventaglio
                        const double scale=1.0)      // scala
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la scala (numero di pips per barra)
```

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare la scala! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia la direzione del trend del Ventaglio di Gann  |
//+------------------------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,        // ID del chart
                            const string name="GannFan",    // nome del ventaglio
                            const bool   direction=true,    // direzione del trend
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la direzione del trend del Ventaglio di Gann
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare la direzione del trend! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| La funzione rimuove il Ventaglio di Gann dal chart                                 |
//+------------------------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,     // ID del chart
                   const string name="GannFan",    // nome del ventaglio
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il Ventaglio di Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il \"Ventaglio di Gann\"! Error code = ",GetL
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio degli del Ventaglio di Gann ed imposta
//| valori di default per quelli vuoti |
```

```
//+------------------------------------------------------------------------------+
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio del Ventaglio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
```

```
//--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il Ventaglio di Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crea il Ventaglio di Gann
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio del ventaglio
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- sposta il primo punto di ancoraggio, verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1<accuracy-1)
         p1+=1;
      //--- sposta il punto
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- cambia la direzione del trend del Ventaglio in discendente
   GannFanDirectionChange(0,InpName,true);
//--- redisegna il chart
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il ventaglio dal chart
   GannFanDelete(0,InpName);
```

```
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_GANNGRID

Griglia di Gann.

Nota

Per le Griglie di Gann, è possibile specificare il tipo di trend dall'enumerazione ENUM_GANN_DIRECTION. Regolando il valore di scala (OBJPROP_SCALE), è possibile cambiare l'angolo di inclinazione delle linee della griglia.

Esempio

Il seguente script crea e sposta la griglia di Gann sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Griglia di Gann\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="GannGrid";        // Nome della Griglia
input int             InpDate1=15;               // data del 1mo punto, %
input int             InpPrice1=25;              // prezzo del 1mo punto, %
input int             InpDate2=35;               // data del 2ndo punto, %
input double          InpScale=3.0;              // Scala
input bool            InpDirection=false;        // Direzione del Trend
```

```
input color           InpColor=clrRed;           // Colore della Griglia
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile delle linee della griglia
input int             InpWidth=1;                // Spessore delle linee ventaglio
input bool            InpBack=false;             // Sottofondo della griglia
input bool            InpSelection=true;         // Evidenzia movimento
input bool            InpHidden=true;            // Nascosto nella lista oggetti
input long            InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea la Griglia di Gann |
//+------------------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // ID del chart
                      const string          name="ArrowBuy",   // nome della griglia
                      const int             sub_window=0,      // indice sottofinestra
                      datetime              time1=0,           // orario del primo pun
                      double                price1=0,          // prezzo del primo pun
                      datetime              time2=0,           // orario del secondo p
                      const double          scale=1.0,         // scala
                      const bool            direction=true,    // direzione del trend
                      const color           clr=clrRed,        // colore della griglia
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee de
                      const int             width=1,           // spessore delle linee
                      const bool            back=false,        // in sottofondo
                      const bool            selection=true,    // evidenzia movimento
                      const bool            hidden=true,       // nascosta nella lista
                      const long            z_order=0)         // priorità per il clic
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea la Griglia di Gann dalle coordinate fornite
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare la \"Griglia di Gann\"! Error code = ",GetLastErr
      return(false);
     }
//--- cambia la scala (numero di pips per barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- cambia la direzione del trend del Ventaglio di Gann (true - discendente, false -
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- imposta il colore della griglia
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della visualizzazione delle linee della griglia
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee della griglia
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

```
//--- abilita (true) o disabilita (false) la modalità di evidenziazione delle griglie
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio della Griglia di Gann  |
//+------------------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,      // ID del chart
                         const string name="ArrowBuy", // nome della griglia
                         const int    point_index=0,   // indice del punto di ancorag
                         datetime     time=0,          // coordinate tempo, del punto
                         double       price=0)         // coordinate di prezzo del pun
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta i punti di ancoraggio della griglia
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia la Scala della Griglia di Gann                                        |
//+------------------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,      // ID del chart
                         const string name="GannGrid", // griglie
                         const double scale=1.0)       // scala
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la scala (numero di pips per barra)
```

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare la scala! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia la direzione del trend delle Griglie di Gann      |
//+------------------------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,       // ID del chart
                             const string name="ArrowBuy", // nome della griglia
                             const bool   direction=true,  // direzione del trend
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la direzione del trend delle Griglie di Gann
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare la direzione del trend! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| La funzione rimuove il Ventaglio di Gann dal chart                                 |
//+------------------------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,          // ID del chart
                    const string  name="ArrowBuy",    // nome della griglia
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina la Griglia di Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare la \"Griglia di Gann\"! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio degli della Griglia di Gann ed imposta
//| valori di default per quelli vuoti |
```

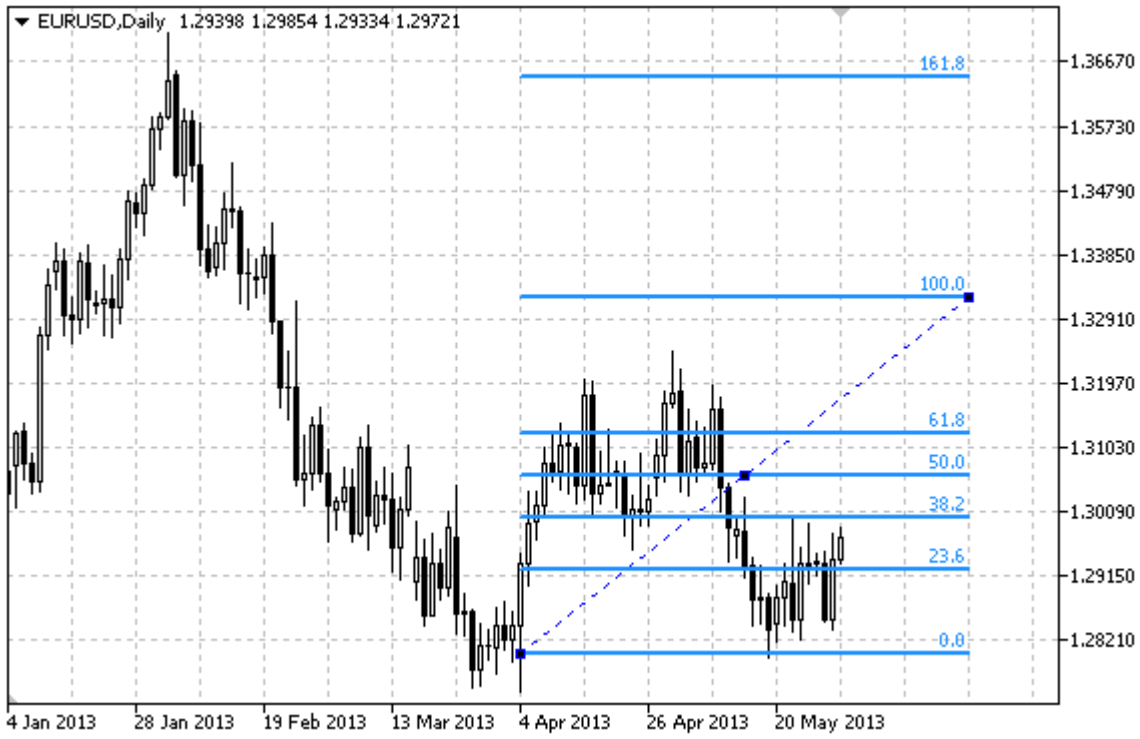
```
//+------------------------------------------------------------------------------+
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio della griglia
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
```

```
//--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare la Griglia di Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- crea la Griglia di Gann
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio della griglia
//---contatore del ciclo
   int v_steps=accuracy/4;
//--- sposta il primo punto di ancoraggio, verticalmente
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int h_steps=bars/4;
//--- sposta il secondo punto di ancoraggio, orizzontalmente
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
```

```
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- cambia la direzione del trend della griglia in discendente
   GannGridDirectionChange(0,InpName,true);
//--- redisegna il chart
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina la griglia dal chart
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_FIBO

Ritracciamenti di Fibonacci.



Nota

Per i "Ritracciamenti di Fibonacci", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente).

È inoltre possibile specificare il numero di linee-livelli, i loro valori ed il colore.

Esempio

Il seguente script crea e sposta i Ritracciamenti di Fibonacci sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Ritracciamenti di Fibonac
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="FiboLevels";      // Nome dell'oggetto
input int             InpDate1=10;               // data del 1mo punto, %
input int             InpPrice1=65;              // prezzo del 1mo punto, %
input int             InpDate2=90;               // data del 2ndo punto, %
input int             InpPrice2=85;              // prezzo del 2ndo punto, %
input color           InpColor=clrRed;           // Colore degli oggetti
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile della linea
input int             InpWidth=2;                // Spessore della linea
input bool            InpBack=false;             // Oggetto di sottofondo
input bool            InpSelection=true;         // Evidenzia movimento
input bool            InpRayLeft=false;          // Continuazione dell'oggetto a sinis
input bool            InpRayRight=false;         // Continuazione dell'oggetto a desti
input bool            InpHidden=true;            // Nascosto nella lista oggetti
input long            InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea i Ritracciamenti di Fibonacci dalle coordinate date                     |
//+------------------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // ID del chart
                      const string          name="FiboLevels", // nome dell'oggetto
                      const int             sub_window=0,      // indice della sottofi
                      datetime              time1=0,           // orario del primo pun
                      double                price1=0,          // prezzo del primo pun
                      datetime              time2=0,           // orario del secondo p
                      double                price2=0,          // prezzo del secondo p
                      const color           clr=clrRed,        // colore dell'oggetto
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // stile del bordo dell
                      const int             width=1,           // spessore della linea
                      const bool            back=false,        // in sottofondo
                      const bool            selection=true,    // evidenza movimento
                      const bool            ray_left=false,    // continuazione dell'o
                      const bool            ray_right=false,   // continuazione dell'o
                      const bool            hidden=true,       // nascosto nella lista
                      const long            z_order=0)         // priorità per il clic
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- Crea i Ritracciamenti di Fibonacci dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare i \"Ritracciamenti di Fibonacci\"! Error code = "
      return(false);
     }
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
```

```
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il numero di livelli ed i loro parametri                             |
//+------------------------------------------------------------------------------+
bool FiboLevelsSet(int             levels,               // numero di linee livello
                   double          &values[],            // valore delle linee livello
                   color           &colors[],            // colore delle linee livello
                   ENUM_LINE_STYLE &styles[],            // stile delle linee livello
                   int             &widths[],            // spessore delle linee livel
                   const long      chart_ID=0,           // ID del chart
                   const string    name="FiboLevels", // nome dell'oggetto
  {
//--- verifica la grandezza dell'array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) || levels!=ArraySize(widt
     {
      Print(__FUNCTION__,": la lungezza dell'array non corrisponde al numero di livel
      return(false);
     }
//--- set the number of levels
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- imposta le proprietà dei livelli nel loop
   for(int i=0;i<levels;i++)
     {
      //--- valore dei livelli
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- colore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- stile dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- spessore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrizione dei livelli
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- esecuzione avvenuta
```

```
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio dei Ritracciamenti di Fibonacci                       |
//+------------------------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,         // ID del chart
                           const string name="FiboLevels", // nome dell'oggetto
                           const int    point_index=0,     // indice del punto di anco
                           datetime     time=0,            // coordinate tempo, del pu
                           double       price=0)           // coordinate di prezzo del
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina i Ritracciamenti di Fibonacci                                              |
//+------------------------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,         // ID del chart
                      const string name="FiboLevels", // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare i \"Ritracciamenti di Fibonacci\"! Error code
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio dei Ritracciamenti di Fibonacci ed impo
//| valori di default per quelli vuoti                                                 |
```

```
//+------------------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se il prezzo del secondo punto non è impostato, avrà un valore Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del primo punto non è impostato, lo sposta di 200 punti sotto al se
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio dei Ritracciament
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

```
      {
       Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
       return;
      }
//--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare i Ritracciamenti di Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea un oggetto
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
      {
       return;
      }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio
//---contatore del ciclo
   int v_steps=accuracy*2/5;
//--- move the first anchor point
   for(int i=0;i<v_steps;i++)
      {
       //--- usa il seguente valore
       if(p1>1)
          p1-=1;
       //--- sposta il punto
       if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
          return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
          return;
       //--- ridisegna il chart
       ChartRedraw();
      }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy*4/5;
//--- sposta il secondo punto di ancoraggio
```

```
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p2>1)
         p2-=1;
      //--- sposta il punto
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'oggetto dal chart
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_FIBOTIMES

Fibonacci Time Zones (zone temporali).

Nota

Per le"Fibonacci Time Zones", è possibile specificare il numero di linee-livello, il loro valore e colore.

Esempio

Il seguente script crea e sposta le Fibonacci Time Zones sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Fibonacci Time Zones\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description "finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="FiboTimes";       // Nome dell'oggetto
input int             InpDate1=10;               // data del 1mo punto, %
input int             InpPrice1=45;              // prezzo del 1mo punto, %
input int             InpDate2=20;               // data del 2ndo punto, %
input int             InpPrice2=55;              // prezzo del 2ndo punto, %
input color           InpColor=clrRed;           // Colore degli oggetti
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile della linea
input int             InpWidth=2;                // Spessore della linea
input bool            InpBack=false;             // Oggetto di sottofondo
```

```
input bool            InpSelection=true;         // Evidenzia movimento
input bool            InpHidden=true;            // Nascosto nella lista oggetti
input long            InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea le Fibonacci Time Zones dalle coordinate fornite                        |
//+------------------------------------------------------------------------------+
bool FiboTimesCreate(const long            chart_ID=0,        // ID del chart
                     const string          name="FiboTimes",  // nome dell'oggetto
                     const int             sub_window=0,      // indice sottofinestra
                     datetime              time1=0,           // orario del primo punt
                     double                price1=0,          // prezzo del primo punt
                     datetime              time2=0,           // orario del secondo pu
                     double                price2=0,          // prezzo del secondo pu
                     const color           clr=clrRed,        // colore dell'oggetto
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // stile del bordo della
                     const int             width=1,           // spessore della linea
                     const bool            back=false,        // in sottofondo
                     const bool            selection=true,    // evidenza movimento
                     const bool            hidden=true,       // nascosto nella lista
                     const long            z_order=0)         // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea le Fibonacci Time Zones dalle coordinate fornite
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare le \"Fibonacci Time Zones\"! Error code = ",GetLa
      return(false);
     }
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il numero di livelli ed i loro parametri                             |
//+------------------------------------------------------------------------------+
bool FiboTimesLevelsSet(int             levels,               // numero di linee livel
                        double          &values[],            // valore delle linee li
                        color           &colors[],            // colore delle linee li
                        ENUM_LINE_STYLE &styles[],            // stile delle linee liv
                        int             &widths[],            // spessore delle linee
                        const long      chart_ID=0,           // ID del chart
                        const string    name="FiboTimes",     // nome dell'oggetto
  {
//--- verifica la grandezza dell'array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la lungezza dell'array non corrisponde al numero di livell
      return(false);
     }
//--- set the number of levels
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- imposta le proprietà dei livelli nel loop
   for(int i=0;i<levels;i++)
     {
      //--- valore dei livelli
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- colore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- stile dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- spessore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrizione dei livelli
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio delleFibonacci Time Zones                       |
//+------------------------------------------------------------------------------+
bool FiboTimesPointChange(const long   chart_ID=0,       // ID del chart
                          const string name="FiboTimes", // nome dell'oggetto
                          const int    point_index=0,    // indice del punto di ancora
                          datetime     time=0,           // coordinate orarie del punt
                          double       price=0)          // coordinate di prezzo del p
```

```
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina le Fibonacci Time Zones                                                    |
//+------------------------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,       // ID del chart
                     const string name="FiboTimes", // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare le \"Fibonacci Time Zones\"! Error code = ",G
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori delle Fibonacci Time zones ed                                   |
//| imposta i valore di default per quelle vuote                                       |
//+------------------------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

```
//--- se l'orario del secondo punto non è impostato, è posizionato 2 barre meno dalla
   if(!time2)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 3 barre
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- imposta il primo punto 2 barre a sinistra dalla seconda
      time2=temp[0];
     }
//--- se il prezzo del secondo punto non è impostato, è uguale a quello del primo punt
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio delle Fibonacci T
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
```

```
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare le Fibonacci Time Zones
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea un oggetto
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio
//---contatore del ciclo
   int h_steps=bars*2/5;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d2<bars-1)
         d2+=1;
      //--- sposta il punto
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   h_steps=bars*3/5;
//--- move the first anchor point
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d1<bars-1)
         d1+=1;
      //--- sposta il punto
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
```

```
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'oggetto dal chart
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_FIBOFAN

Ventagli di Fibonacci.

Nota

Per i "Ventagli di Fibonacci", è possibile specificare il numero di linee-livello, il loro valore e colore.

Esempio

Il seguente script crea e sposta il Ventaglio di Fibonacci sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Ventaglio di Fibonacci\"
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="FiboFan";          // Nome dell'oggetto
input int             InpDate1=10;                // data del 1mo punto, %
input int             InpPrice1=25;               // prezzo del 1mo punto, %
input int             InpDate2=30;                // data del 2ndo punto, %
input int             InpPrice2=50;               // prezzo del 2ndo punto, %
input color           InpColor=clrRed;            // Colore della linea Ventaglio
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile della linea
input int             InpWidth=2;                 // Spessore della linea
```

```
input bool              InpBack=false;              // Oggetto di sottofondo
input bool              InpSelection=true;          // Evidenzia movimento
input bool              InpHidden=true;             // Nascosto nella lista oggetti
input long              InpZOrder=0;                // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il Ventaglio di Fibonacci dalle coordinate fornite                      |
//+------------------------------------------------------------------------------+
bool FiboFanCreate(const long             chart_ID=0,        // ID del chart
                   const string           name="FiboFan",    // nome del ventaglio
                   const int              sub_window=0,      // indice sottofinestra
                   datetime               time1=0,           // orario del primo punto
                   double                 price1=0,          // prezzo del primo punto
                   datetime               time2=0,           // orario del secondo punt
                   double                 price2=0,          // prezzo del secondo punt
                   const color            clr=clrRed,        // colore della linea vent
                   const ENUM_LINE_STYLE  style=STYLE_SOLID, // stile della linea venta
                   const int              width=1,           // spessore della linea ve
                   const bool             back=false,        // in sottofondo
                   const bool             selection=true,    // evidenzia movimento
                   const bool             hidden=true,       // nascosto nella lista og
                   const long             z_order=0)         // priorità per il click d
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il Ventaglio di Fibonacci dalle coordinate fornite
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il \"Ventaglio di Fibonacci\"! Error code = ",Get
      return(false);
     }
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del venglio per lo
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

```
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il numero di livelli ed i loro parametri                             |
//+------------------------------------------------------------------------------+
bool FiboFanLevelsSet(int             levels,               // numero di linee livello
                      double          &values[],            // valore delle linee live
                      color           &colors[],            // colore delle linee live
                      ENUM_LINE_STYLE &styles[],            // stile delle linee livel
                      int             &widths[],            // spessore delle linee li
                      const long      chart_ID=0,           // ID del chart
                      const string    name="FiboFan",       // nome del ventaglio
  {
//--- verifica la grandezza dell'array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la lungezza dell'array non corrisponde al numero di livell
      return(false);
     }
//--- set the number of levels
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- imposta le proprietà dei livelli nel loop
   for(int i=0;i<levels;i++)
     {
      //--- valore dei livelli
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- colore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- stile dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- spessore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrizione dei livelli
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio del Ventaglio di Fibonacci                      |
//+------------------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // ID del chart
                        const string name="FiboFan", // nome del ventaglio
                        const int    point_index=0,  // indice del punto di ancoraggio
                        datetime     time=0,         // coordinate del punto di ancora
```

```
                        double       price=0)        // coordinate prezzo, del punto d
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il ventaglio di Fibonacci                                                  |
//+------------------------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // ID del chart
             const string          name="FiboFan",    // nome del ventaglio
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il ventaglio
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il \"Ventaglio di Fibonacci\"! Error code = "
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio del Ventaglio di Fibonacci ed imposta i
//| valori di default per quelli vuoti                                                 |
//+------------------------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se il prezzo del secondo punto non è impostato, avrà un valore Bid
   if(!price2)
```

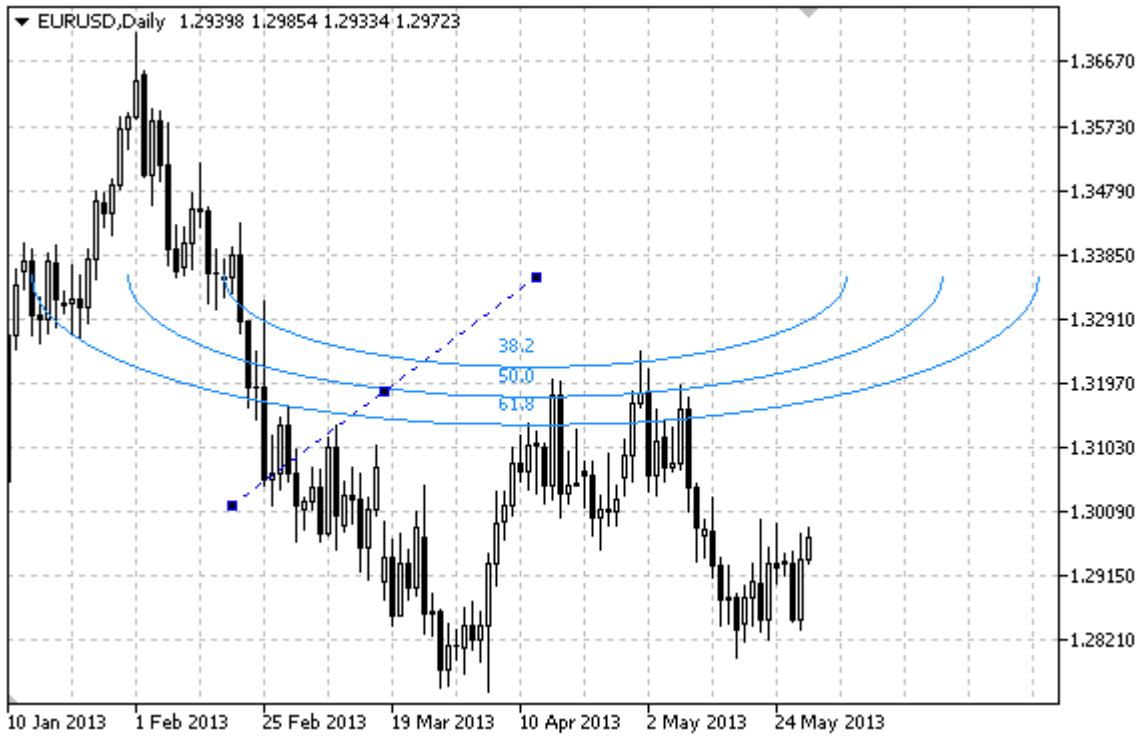
```
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del primo punto non è impostato, lo sposta di 200 punti sotto al se
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio del Ventaglio di
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
```

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il Ventaglio di Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea un oggetto
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio del ventaglio
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- move the first anchor point
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1<accuracy-1)
         p1+=1;
      //--- sposta il punto
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int h_steps=bars/4;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d2<bars-1)
         d2+=1;
      //--- sposta il punto
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
```

```
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'oggetto dal chart
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_FIBOARC

Archi di Fibonacci.



Nota

Per gli "Archi di Fibonacci", è possibile specificare la modalità di visualizzazione dell'intera ellisse. Il raggio di curvatura può essere specificato modificando la scala e le coordinate dei punti di ancoraggio.

È inoltre possibile specificare il numero di linee-livelli, i loro valori ed il colore.

Esempio

Il seguente script crea e sposta gli Archi di Fibonacci Arcs sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Archi di Fibonacci\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="FiboArc";          // Nome dell'oggetto
input int             InpDate1=25;                // data del 1mo punto, %
input int             InpPrice1=25;               // prezzo del 1mo punto, %
input int             InpDate2=35;                // data del 2ndo punto, %
input int             InpPrice2=55;               // prezzo del 2ndo punto, %
```

```
input double           InpScale=3.0;               // Scala
input bool             InpFullEllipse=true;        // Forma degli archi
input color            InpColor=clrRed;            // Colore della linea
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile della linea
input int              InpWidth=2;                 // Spessore della linea
input bool             InpBack=false;              // Oggetto di sottofondo
input bool             InpSelection=true;          // Evidenzia movimento
input bool             InpHidden=true;             // Nascosto nella lista oggetti
input long             InpZOrder=0;                // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Create Fibonacci Arcs by the given coordinates                               |
//+------------------------------------------------------------------------------+
bool FiboArcCreate(const long            chart_ID=0,          // ID del chart
                   const string          name="FiboArc",      // nome dell'oggetto
                   const int             sub_window=0,        // indice sottofinestra
                   datetime              time1=0,             // orario del primo punto
                   double                price1=0,            // prezzo del primo punto
                   datetime              time2=0,             // orario del secondo pun
                   double                price2=0,            // prezzo del secondo pun
                   const double          scale=1.0,           // scala
                   const bool            full_ellipse=false, // forma degli archi
                   const color           clr=clrRed,          // colore della linea
                   const ENUM_LINE_STYLE style=STYLE_SOLID,  // stile della linea
                   const int             width=1,             // spessore della linea
                   const bool            back=false,          // in sottofondo
                   const bool            selection=true,      // evidenzia movimento
                   const bool            hidden=true,         // nascosto nella lista
                   const long            z_order=0)           // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea gli Archi di Fibonacci dalle coordinate fornite|
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare gli \"Archi di Fibonacci\"! Error code = ",GetLas
      return(false);
     }
//--- imposta la scala
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- imposta la visualizzazione degli archi come un'ellisse completa (true) o una met
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- larghezza della linea
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia degli archi per lo
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il numero di livelli ed i loro parametri                             |
//+------------------------------------------------------------------------------+
bool FiboArcLevelsSet(int              levels,               // numero di linee livello
                      double           &values[],            // valore delle linee live
                      color            &colors[],            // colore delle linee live
                      ENUM_LINE_STYLE &styles[],             // stile delle linee livel
                      int              &widths[],            // spessore delle linee li
                      const long       chart_ID=0,           // ID del chart
                      const string     name="FiboArc",       // nome dell'oggetto
  {
//--- verifica la grandezza dell'array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la lungezza dell'array non corrisponde al numero di livell
      return(false);
     }
//--- set the number of levels
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- imposta le proprietà dei livelli nel loop
   for(int i=0;i<levels;i++)
     {
      //--- valore dei livelli
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- colore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- stile dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- spessore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrizione dei livelli
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
```

```
      }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio degli Archi di Fibonacci                              |
//+------------------------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,     // ID del chart
                        const string name="FiboArc", // nome dell'oggetto
                        const int    point_index=0,  // indice del punto di ancoraggio
                        datetime     time=0,         // coordinate del punto di ancora
                        double       price=0)        // coordinate prezzo, del punto d
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina gli Archi di Fibonacci                                                     |
//+------------------------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,     // ID del chart
                   const string name="FiboArc", // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare gli \"Archi di Fibonacci\"! Error code = ",Ge
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
```

```
//| Controlla i valori dei punti di ancoraggio degli Archi di Fibonacci ed imposta i
//| valori di default per quelli vuoti                                               |
//+----------------------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- se l'orario del secondo punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se il prezzo del secondo punto non è impostato, avrà un valore Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del primo punto non è impostato, è posizionato 9 barre meno dalla se
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del primo punto non è impostato, lo sposta di 300 punti sotto al se
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+----------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                           |
//+----------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio degli Archi di Fi
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
```

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare gli Archi di Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea un oggetto
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio
//---contatore del ciclo
   int v_steps=accuracy/5;
//--- move the first anchor point
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1<accuracy-1)
         p1+=1;
      //--- sposta il punto
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
```

```
   int h_steps=bars/5;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d2<bars-1)
         d2+=1;
      //--- sposta il punto
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'oggetto dal chart
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_FIBOCHANNEL

Canale di Fibonacci.

Nota

Per il "Canale di Fibonacci", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT ed OBJPROP_RAY_LEFT rispettivamente).

È inoltre possibile specificare il numero di linee-livelli, i loro valori ed il colore.

Esempio

Il seguente script crea e sposta il Canale di Fibonacci sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Canale di Fibonacci\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="FiboChannel";     // Nome del canale
input int             InpDate1=20;               // data del 1mo punto, %
input int             InpPrice1=10;              // prezzo del 1mo punto, %
input int             InpDate2=60;               // data del 2ndo punto, %
input int             InpPrice2=30;              // prezzo del 2ndo punto, %
input int             InpDate3=20;               // data del 3zo punto, %
```

```
input int              InpPrice3=25;              // prezzo del 3zo punto, %
input color            InpColor=clrRed;           // Colore del canale
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Stile delle linee del canale
input int              InpWidth=2;                // Spessore delle linee del canale
input bool             InpBack=false;             // Canale di sottofondo
input bool             InpSelection=true;         // Evidenzia movimento
input bool             InpRayLeft=false;          // Continuazione del canale a sinistr
input bool             InpRayRight=false;         // Continuazione del canale a destra
input bool             InpHidden=true;            // Nascosto nella lista oggetti
input long             InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------------+
//| Crea il Canale di Fibonacci dalle coordinate date                                  |
//+------------------------------------------------------------------------------------+
bool FiboChannelCreate(const long             chart_ID=0,         // ID del chart
                       const string           name="FiboChannel", // nome del canale
                       const int              sub_window=0,       // indice sottofinest
                       datetime               time1=0,            // orario del primo p
                       double                 price1=0,           // prezzo del primo p
                       datetime               time2=0,            // orario del second
                       double                 price2=0,           // prezzo del second
                       datetime               time3=0,            // orario del terzo p
                       double                 price3=0,           // prezzo del terzo p
                       const color            clr=clrRed,         // colore del canale
                       const ENUM_LINE_STYLE  style=STYLE_SOLID,  // stile delle linee
                       const int              width=1,            // spesso delle linee
                       const bool             back=false,         // in sottofondo
                       const bool             selection=true,     // evidenzia moviment
                       const bool             ray_left=false,     // continua del canal
                       const bool             ray_right=false,    // continua del canal
                       const bool             hidden=true,        // nascosto nella lis
                       const long             z_order=0)          // priorità per il cl
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il canale dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il \"Canale di Fibonacci\"! Error code = ",GetLas
      return(false);
     }
//--- imposta il colore del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee del canale
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

```
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione del display del
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il numero di livelli ed i loro parametri                             |
//+------------------------------------------------------------------------------+
bool FiboChannelLevelsSet(int             levels,          // numero di linee liv
                          double          &values[],       // valore delle linee
                          color           &colors[],       // colore delle linee
                          ENUM_LINE_STYLE &styles[],       // stile delle linee
                          int             &widths[],       // spessore delle line
                          const long      chart_ID=0,      // ID del chart
                          const string    name="FiboChannel")   // nome dell'oggetto
  {
//--- verifica la grandezza dell'array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la lungezza dell'array non corrisponde al numero di livell
      return(false);
     }
//--- set the number of levels
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- imposta le proprietà dei livelli nel loop
   for(int i=0;i<levels;i++)
     {
      //--- valore dei livelli
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- colore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- stile dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- spessore dei livelli
```

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrizione dei livelli
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio del Canale di Fibonacci                               |
//+------------------------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,          // ID del chart
                            const string name="FiboChannel", // nome del canale
                            const int    point_index=0,      // indice del punto di ar
                            datetime     time=0,             // coordinate tempo, del
                            double       price=0)            // coordinate prezzo, del
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il canale                                                                  |
//+------------------------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,          // ID del chart
                       const string name="FiboChannel", // nome del canale
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il canale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il \"Canale di Fibonacci\"! Error code = ",Ge
      return(false);
     }
//--- esecuzione avvenuta
```

```
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio del Canale di Fibonacci ed imposta i|
//| valori di default per quelli vuoti                                                 |
//+------------------------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- se l'orario del secondo(destro) punto non è impostato, sarà sulla barra corrente
   if(!time2)
      time2=TimeCurrent();
//--- se il prezzo del secondo punto non è impostato, avrà un valore Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del primo(sinistro) punto non è impostato, è posizionato 9 barre mer
   if(!time1)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del primo punto non è impostato, lo sposta di 300 punti in più risp
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se l'orario del terzo punto non è impostato, esso coincide con quello del primo
   if(!time3)
      time3=time1;
//--- se il prezzo del terzo punto non è impostato, è uguale a quello del secondo punt
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

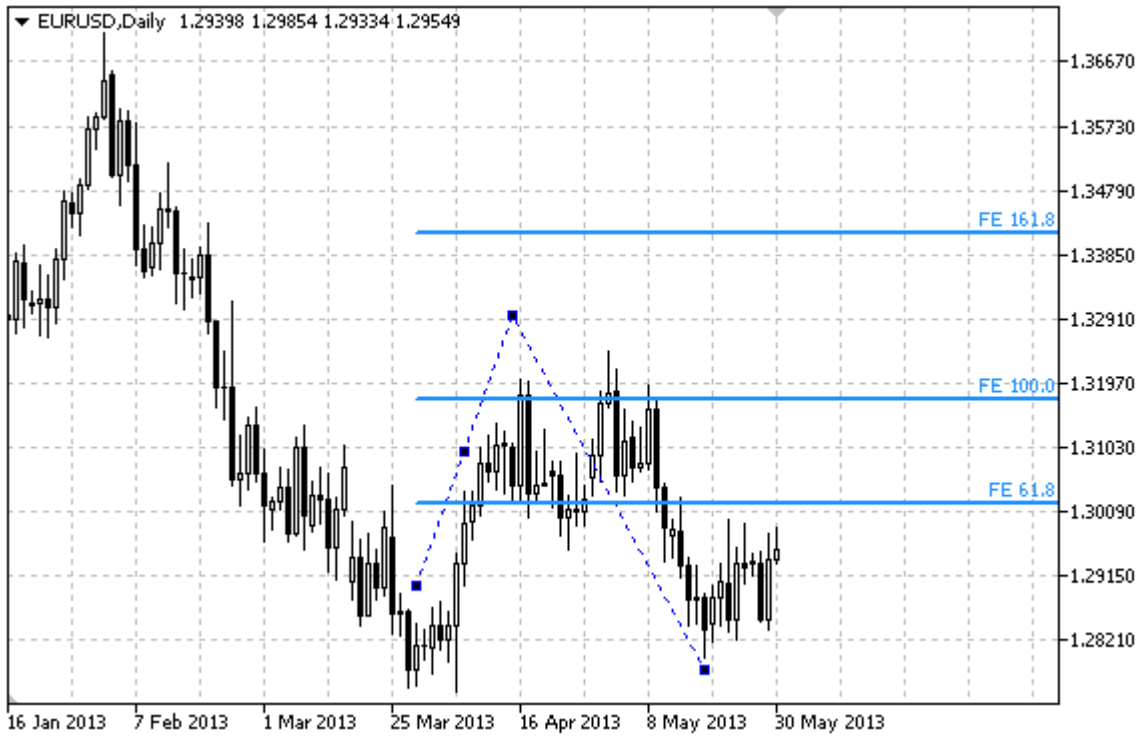
```
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e modificare le coordinate dei punti di ancoraggio del canale"
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il canale
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crea il Canale di Fibonacci
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio del canale
//---contatore del ciclo
   int h_steps=bars/10;
//--- move the first anchor point
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d1>1)
         d1-=1;
```

```
      //--- sposta il punto
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int v_steps=accuracy/10;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p2>1)
         p2-=1;
      //--- sposta il punto
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy/15;
//--- sposta il terzo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p3<accuracy-1)
         p3+=1;
      //--- sposta il punto
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
```

```
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il canale dal chart
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_EXPANSION

Espansioni di Fibonacci.



Nota

Per le "Espansioni di Fibonacci", è possibile specificare la modalità della sua continuazione a destra e/o a sinistra (proprietà OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT rispettivamente).

È inoltre possibile specificare il numero di linee-livelli, i loro valori ed il colore.

Esempio

Il seguente script crea e sposta le Espansioni di Fibonacci sul grafico. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna l'oggetto grafico \"Espansioni di Fibonacci\"
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="FiboExpansion";    // Nome dell'oggetto
input int             InpDate1=10;                // data del 1mo punto, %
input int             InpPrice1=55;               // prezzo del 1mo punto, %
input int             InpDate2=30;                // data del 2ndo punto, %
input int             InpPrice2=10;               // prezzo del 2ndo punto, %
input int             InpDate3=80;                // data del 3zo punto, %
```

```
input int              InpPrice3=75;              // prezzo del 3zo punto, %
input color            InpColor=clrRed;           // Colore degli oggetti
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Stile delle linee
input int              InpWidth=2;                // Spessore delle linee
input bool             InpBack=false;             // Oggetto di sottofondo
input bool             InpSelection=true;         // Evidenzia movimento
input bool             InpRayLeft=false;          // Continuazione dell'oggetto a sinis
input bool             InpRayRight=false;         // Continuazione dell'oggetto a dest
input bool             InpHidden=true;            // Nascosto nella lista oggetti
input long             InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------------+
//| Crea le Estensioni di Fibonacci dalle coordinate date                              |
//+------------------------------------------------------------------------------------+
bool FiboExpansionCreate(const long             chart_ID=0,           // ID del chart
                         const string           name="FiboExpansion", // nome del canal
                         const int              sub_window=0,         // indice sottof
                         datetime               time1=0,              // orario del pr
                         double                 price1=0,             // prezzo del pr
                         datetime               time2=0,              // orario del se
                         double                 price2=0,             // prezzo del se
                         datetime               time3=0,              // orario del te
                         double                 price3=0,             // prezzo del te
                         const color            clr=clrRed,           // colore dell'o
                         const ENUM_LINE_STYLE  style=STYLE_SOLID,    // stile delle l
                         const int              width=1,              // spessore dell
                         const bool             back=false,           // in sottofondo
                         const bool             selection=true,       // evidenzia mov
                         const bool             ray_left=false,       // continuazione
                         const bool             ray_right=false,      // continuazione
                         const bool             hidden=true,          // nascosto nell
                         const long             z_order=0)            // priorità per
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- resetta il valore dell' errore
   ResetLastError();
//--- Crea le Estensioni di Fibonacci per le coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": fallimento nel creare le \"Estensioni di Fibonacci\"! Error code = ",Ge
      return(false);
     }
//--- imposta il colore dell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

```
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- abilita (true) o disabilita (false) la modalità di continuazione della visualizz
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il numero di livelli ed i loro parametri                             |
//+------------------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int             levels,               // numero di linee
                            double          &values[],            // valore delle line
                            color           &colors[],            // colore delle line
                            ENUM_LINE_STYLE &styles[],            // stile delle linee
                            int             &widths[],            // spessore delle li
                            const long      chart_ID=0,           // ID del chart
                            const string    name="FiboExpansion") // nome dell'oggetto
  {
//--- verifica la grandezza dell'array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la lungezza dell'array non corrisponde al numero di livell
      return(false);
     }
//--- set the number of levels
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- imposta le proprietà dei livelli nel loop
   for(int i=0;i<levels;i++)
     {
      //--- valore dei livelli
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- colore dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- stile dei livelli
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- spessore dei livelli
```

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrizione dei livelli
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio delle Espansioni di Fibonacci                   |
//+------------------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,           // ID del chart
                              const string name="FiboExpansion", // nome oggetto
                              const int    point_index=0,        // indice punto di ar
                              datetime     time=0,               // coordinate orarie
                              double       price=0)              // coorinate prezzo,
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina le Espansioni di Fibonacci                                           |
//+------------------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // ID del chart
                         const string name="FiboExpansion") // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare le \"Espansioni di Fibonacci\"! Error code =
      return(false);
     }
//--- esecuzione avvenuta
```

```
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio delle Espansioni di Fibonacci ed impost
//| valori di default per quelli vuoti                                                 |
//+------------------------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- se l'orario del terzo (destro) punto non è impostato, sarà sulla barra corrente
   if(!time3)
      time3=TimeCurrent();
//--- if the third point's price is not set, it will have Bid value
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del primo(sinistro) punto non è impostato, è localizzato 9 barre a s
//--- array per ricevere l'orario di apertura delle ultime 10 barre
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- imposta il primo punto 9 barre a sinistra dalla seconda
      time1=temp[0];
     }
//--- se il prezzo del primo punto non è impostato, è uguale a quello del terzo punto
   if(!price1)
      price1=price3;
//--- se l'orario del secondo punto non è impostato, è posizionato 7 barre meno dalla
   if(!time2)
      time2=temp[2];
//--- se il prezzo del secondo punto non è impostato, lo sposta di 250 punti in meno r
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
```

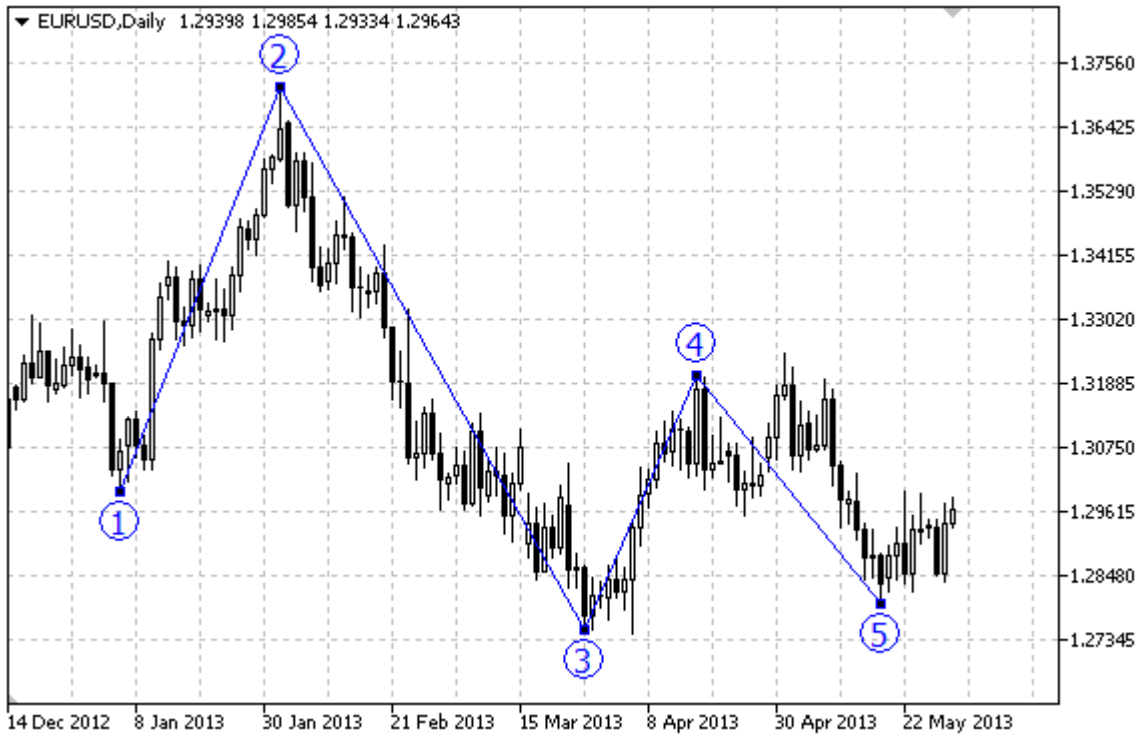
```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio dell'oggetto
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare le Espansioni di Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crea le Espansioni di Fibonacci
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio
//---contatore del ciclo
   int v_steps=accuracy/10;
//--- move the first anchor point
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1>1)
```

```
         p1-=1;
      //--- sposta il punto
      if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy/2;
//--- sposta il terzo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p3>1)
         p3-=1;
      //--- sposta il punto
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy*4/5;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p2<accuracy-1)
         p2+=1;
      //--- sposta il punto
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
```

```
   Sleep(1000);
//--- elimina l'oggetto dal chart
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ELLIOTWAVE5

Onda Movente di Elliott.



Nota

Per l' "Onda Movente di Elliott", è possibile abilitare /disabilitare la modalità di connessione dei punti per righe (OBJPROP_DRAWLINES property), così come impostare il livello di posizione dell'onda (dall'enumarazione ENUM_ELLIOT_WAVE_DEGREE).

Esempio

Il seguente scripe crea e sposta l'onda movente di Elliot sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Onda Movente di Elliott\"
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description "finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string                InpName="ElliotWave5";   // Nome Oggetto
input int                   InpDate1=10;             // data del 1mo punto, %
input int                   InpPrice1=90;            // prezzo del 1mo punto, %
input int                   InpDate2=20;             // data del secondo punto, %
input int                   InpPrice2=40;            // prezzo del 2ndo punto, %
input int                   InpDate3=30;             // data del 3zo punto, %
```

```
input int                      InpPrice3=60;            // prezzo del 3zo punto, %
input int                      InpDate4=40;             // data del 4to punto, %
input int                      InpPrice4=10;            // prezzo del 4to punto, %
input int                      InpDate5=60;             // data del 5to punto, %
input int                      InpPrice5=40;            // prezzo del 5to punto, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Livello
input bool                     InpDrawLines=true;       // Mostra le linee
input color                    InpColor=clrRed;         // Colore delle linee
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Stile delle linee
input int                      InpWidth=2;              // Spessore delle linee
input bool                     InpBack=false;           // Oggetti in sottofondo
input bool                     InpSelection=true;       // Evidenzia movimento
input bool                     InpHidden=true;          // Nascosto nella lista oggetti
input long                     InpZOrder=0;             // Priorità per il click del mo
//+------------------------------------------------------------------------------+
//| Crea l' "Onda Movente di Elliott" dalle coordinate date        |
//+------------------------------------------------------------------------------+
bool ElliotWave5Create(const long                   chart_ID=0,               // ID de
                       const string                 name="ElliotWave5",       // nome
                       const int                    sub_window=0,             // indic
                       datetime                     time1=0,                  // orari
                       double                       price1=0,                 // prezz
                       datetime                     time2=0,                  // orari
                       double                       price2=0,                 // prezz
                       datetime                     time3=0,                  // orari
                       double                       price3=0,                 // prezz
                       datetime                     time4=0,                  // orari
                       double                       price4=0,                 // prezz
                       datetime                     time5=0,                  // orari
                       double                       price5=0,                 // prezz
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // gradi
                       const bool                   draw_lines=true,          // mostr
                       const color                  clr=clrRed,               // color
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee
                       const int                    width=1,                  // spess
                       const bool                   back=false,               // in sc
                       const bool                   selection=true,           // evide
                       const bool                   hidden=true,              // nasco
                       const long                   z_order=0)                // prior
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- resetta il valore dell' errore
   ResetLastError();
//--- Crea l' "Onda Movente di Elliott" dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

```
            ": fallimento nel creare l' \"Onda Movente di Elliott\"! Error code = ",Ge
      return(false);
     }
//--- imposta grado (grandezza onda)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- abilita (true) o disabilita (false) la modalità di visualizzazione delle righe
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- imposta il colore dell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio dell' Onda Movente di Elliott |
//+------------------------------------------------------------------------------+
bool ElliotWave5PointChange(const long   chart_ID=0,          // ID del chart
                            const string name="ElliotWave5", // nome dell'oggetto
                            const int    point_index=0,      // indice del punto di an
                            datetime     time=0,             // coordinate tempo, del
                            double       price=0)            // coordinate prezzo, del
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
```

```
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina l'Onda Movente di Elliott                                                  |
//+------------------------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,         // ID del chart
                       const string name="ElliotWave5") // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l' \"Onda Movente di Elliott\"! Error code =
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Imposta i valori del punto di ancoraggio dell' Onda Movente di Elliott     |
//| imposta i valore di default per quelle vuote                                       |
//+------------------------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- array per ricevere l'orario di apertura delle ultime 10 barre
   datetime temp[];
   ArrayResize(temp,10);
//--- ricezione dati
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- riceve il valore di un punto sul corrente chart
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se l'orario del primo punto non è impostato, sarà 0 barre dietro dall'ultima ba
   if(!time1)
      time1=temp[0];
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, sarà 7 barre dietro dall'ultima
   if(!time2)
      time2=temp[2];
```

```
//--- se il prezzo del secondo punto non è impostato, lo sposta di 300 punti in meno r
   if(!price2)
      price2=price1-300*point;
//--- se l'orario del terzo punto non è impostato, sarà 5 barre dietro dall'ultima bar
   if(!time3)
      time3=temp[4];
//--- se il prezzo del terzo punto non è impostato, lo sposta di 200 punti in meno ris
   if(!price3)
      price3=price1-250*point;
//--- se l'orario del quarto punto non è impostato, sarà 3 barre dietro dall'ultima ba
   if(!time4)
      time4=temp[6];
//--- se il prezzo del quarto punto non è impostato, lo sposta di 550 punti in meno ri
   if(!price4)
      price3=price1-200*point;
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time5)
      time5=temp[9];
//--- se il prezzo del terzo punto non è impostato, lo sposta di 250 punti in meno ris
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
      InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
      InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio dell'oggetto
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
```

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare l'Onda Movente di Elliott
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int d4=InpDate4*(bars-1)/100;
   int d5=InpDate5*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
   int p4=InpPrice4*(accuracy-1)/100;
   int p5=InpPrice5*(accuracy-1)/100;
//--- Crea l'Onda Movente di Elliott
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,I
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio
//---contatore del ciclo
   int v_steps=accuracy/5;
//--- sposta il quinto punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p5<accuracy-1)
         p5+=1;
      //--- sposta il punto
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
```

```
         return;
       //--- ridisegna il chart
       ChartRedraw();
      }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy/5;
//--- sposta il secondo ed il terzo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa i seguenti valori
      if(p2<accuracy-1)
         p2+=1;
      if(p3>1)
         p3-=1;
      //--- slitta i punti
      if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy*4/5;
//--- sposta il primo ed il quarto punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa i seguenti valori
      if(p1>1)
         p1-=1;
      if(p4<accuracy-1)
         p4+=1;
      //--- slitta i punti
      if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
```

```
      }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'oggetto dal chart
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ELLIOTWAVE3

Onda Correttiva di Elliott.

Nota

Per l' "Onda Correttiva di Elliott", è possibile abilitare /disabilitare la modalità di connessione dei punti per righe (OBJPROP_DRAWLINES property), così come impostare il livello di posizione dell'onda (dall'enumarazione ENUM_ELLIOT_WAVE_DEGREE).

Esempio

Il seguente scripe crea e sposta l'onda correttiva di Elliot sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Onda Correttiva di Elliot
#property description "Le coordinate del punto di ancoraggio sono impostare in percent
#property description "grandezza."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string                InpName="ElliotWave3";   // Nome oggetto
input int                   InpDate1=10;             // data del 1mo punto, %
input int                   InpPrice1=90;            // prezzo del 1mo punto, %
input int                   InpDate2=30;             // data del secondo punto, %
input int                   InpPrice2=10;            // prezzo del 2ndo punto, %
input int                   InpDate3=50;             // data del 3zo punto, %
```

```
input int                      InpPrice3=40;            // prezzo del 3zo punto, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Livello
input bool                     InpDrawLines=true;       // Mostra le linee
input color                    InpColor=clrRed;         // Colore delle linee
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Stile delle linee
input int                      InpWidth=2;              // Spessore delle linee
input bool                     InpBack=false;           // Oggetti in sottofondo
input bool                     InpSelection=true;       // Evidenzia movimento
input bool                     InpHidden=true;          // Nascosto nella lista oggetti
input long                     InpZOrder=0;             // Priorità per il click del mo
//+------------------------------------------------------------------------------------+
//| Crea l' "Onda Correttiva di Elliott" dalle coordinate date                         |
//+------------------------------------------------------------------------------------+
bool ElliotWave3Create(const long                    chart_ID=0,              // ID de
                       const string                  name="ElliotWave3",      // nome
                       const int                     sub_window=0,            // indic
                       datetime                      time1=0,                 // orari
                       double                        price1=0,                // prezz
                       datetime                      time2=0,                 // orari
                       double                        price2=0,                // prezz
                       datetime                      time3=0,                 // orari
                       double                        price3=0,                // prezz
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // gradi
                       const bool                    draw_lines=true,         // mostr
                       const color                   clr=clrRed,              // color
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee
                       const int                     width=1,                 // spess
                       const bool                    back=false,              // in sc
                       const bool                    selection=true,          // evide
                       const bool                    hidden=true,             // nasco
                       const long                    z_order=0)               // prio
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- resetta il valore dell' errore
   ResetLastError();
//--- Crea l' "Onda Correttiva di Elliott" dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l' \"Onda Correttiva di Elliott\"! Error code = "
      return(false);
     }
//--- imposta grado (grandezza onda)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- abilita (true) o disabilita (false) la modalità di visualizzazione delle righe
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- imposta il colore dell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

```
//--- imposta lo stile delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenzia del canale per lo s
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio dell' Onda Correttiva di Elliott               |
//+------------------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,          // ID del chart
                            const string name="ElliotWave3", // nome dell'oggetto
                            const int    point_index=0,      // indice del punto di ar
                            datetime     time=0,             // coordinate tempo, del
                            double       price=0)            // coordinate prezzo, del
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina l'Onda Correttiva di Elliott                                         |
//+------------------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,          // ID del chart
```

```
                        const string name="ElliotWave3") // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l' \"Onda Correttiva di Elliott\"! Error code
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta i valori del punto di ancoraggio dell' Onda Correttiva di Elliott    |
//| ed imposta il valore di default per quelli vuoti                             |
//+------------------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- array per ricevere l'orario di apertura delle ultime 10 barre
   datetime temp[];
   ArrayResize(temp,10);
//--- ricezione dati
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- riceve il valore di un punto sul corrente chart
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se l'orario del primo punto non è impostato, sarà 0 barre dietro dall'ultima ba
   if(!time1)
      time1=temp[0];
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, sarà 5 barre dietro dall'ultima
   if(!time2)
      time2=temp[4];
//--- se il prezzo del secondo punto non è impostato, lo sposta di 300 punti in meno
   if(!price2)
      price2=price1-300*point;
//--- se l'orario del terzo punto non è impostato, sarà 1 barra dietro dall'ultima ba
   if(!time3)
      time3=temp[8];
//--- se il prezzo del terzo punto non è impostato, lo sposta di 300 punti in meno ris
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione di avvio del programma Script                                   |
//+--------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio dell'oggetto
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare l'Onda Correttiva di Elliott
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- Crea l'Onda Correttiva di Elliott
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidde
     {
```

```
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio
//---contatore del ciclo
   int v_steps=accuracy/5;
//--- sposta il terzo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p3<accuracy-1)
         p3+=1;
      //--- sposta il punto
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy*4/5;
//--- sposta il primo ed il secondo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa i seguenti valori
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- slitta i punti
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'oggetto dal chart
```

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_RECTANGLE

Rettangolo.

Nota

Per il rettangolo, la modalità di riempimento con colore può essere impostata utilizzando la proprietà OBJPROP_FILL.

Esempio

Il seguente script crea e sposta il rettangolo sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script crea il rettangolo nel chart."
#property description "Anchor point coordinates are set in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Rectangle"; // Nome del rettangolo
input int             InpDate1=40;         // data del 1mo punto, %
input int             InpPrice1=40;        // prezzo del 1mo punto, %
input int             InpDate2=60;         // data del 2ndo punto, %
input int             InpPrice2=60;        // prezzo del 2ndo punto, %
input color           InpColor=clrRed;     // Colore della linea
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Stile delle linee del rettangolo
```

```
input int              InpWidth=2;          // Spessore delle linee del rettangolo
input bool             InpFill=true;        // Riempie il rettangolo con colore
input bool             InpBack=false;       // Sottofondo rettangolo
input bool             InpSelection=true;   // Evdenzia movimento
input bool             InpHidden=true;      // Nascosto nella lista oggetti
input long             InpZOrder=0;         // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il rettangolo dalle coordinate data                                     |
//+------------------------------------------------------------------------------+
bool RectangleCreate(const long            chart_ID=0,        // ID del chart
                     const string          name="Rectangle",  // nome del rettangolo
                     const int             sub_window=0,      // indice sottofinestra
                     datetime              time1=0,           // orario del primo punt
                     double                price1=0,          // prezzo del primo punt
                     datetime              time2=0,           // orario del secondo pu
                     double                price2=0,          // prezzo del secondo pu
                     const color           clr=clrRed,        // colore del rettangolo
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee del
                     const int             width=1,           // spessore delle linee
                     const bool            fill=false,        // riempie il rettangolo
                     const bool            back=false,        // in sottofondo
                     const bool            selection=true,    // evidenza movimento
                     const bool            hidden=true,       // nascosto nella lista
                     const long            z_order=0)         // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il rettangolo dalle coordinate fornite
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il rettangolo! Error code = ",GetLastError());
      return(false);
     }
//--- imposta il colore del rettangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee del rettangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee del rettangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- abilita (true) o disabilita (false) la modalità di riempimento del rettangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenziazione del rettangolo
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
```

```
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli oc
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio del rettangolo                                 |
//+------------------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,       // ID del chart
                          const string name="Rectangle", // nome del rettangolo
                          const int    point_index=0,    // indice del punto di ancora
                          datetime     time=0,           // coordinate orarie del punt
                          double       price=0)          // coordinate di prezzo del p
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina il rettangolo                                                        |
//+------------------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,       // ID del chart
                     const string name="Rectangle") // nome del rettangolo
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina rettangolo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il rettangolo! Error code = ",GetLastError())
```

```
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio del rettangolo ed imposta         |
//| valori di default per quelli vuoti                                           |
//+------------------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, è posizionato 9 barre meno dalla
   if(!time2)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- imposta il secondo punto, 9 barre in meno dalla prima
      time2=temp[0];
     }
//--- se il prezzo del secondo punto non è impostato, lo sposta di 300 punti in meno
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio del rettangolo
```

```
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il rettangolo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- crea il rettangolo
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio del rettangolo
//---contatore del ciclo
   int h_steps=bars/2;
//--- sposta i punti di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa i seguenti valori
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- slitta i punti
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
```

```
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- sposta i punti di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa i seguenti valori
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- slitta i punti
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il rettangolo dal chart
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_TRIANGLE

Triangolo.

Nota

Per il triangolo, la modalità di riempimento con colore può essere impostata utilizzando la proprietà OBJPROP_FILL.

Esempio

Il seguente script crea e sposta il triangolo sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script crea il triangolo sul chart."
#property description "Anchor point coordinates are set in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Triangle";        // Nome del triangolo
input int             InpDate1=25;               // data del 1mo punto, %
input int             InpPrice1=50;              // prezzo del 1mo punto, %
input int             InpDate2=70;               // data del 2ndo punto, %
input int             InpPrice2=70;              // prezzo del 2ndo punto, %
input int             InpDate3=65;               // data del 3zo punto, %
input int             InpPrice3=20;              // prezzo del 3zo punto, %
```

```
input color           InpColor=clrRed;           // Colore del triangolo
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile delle linee del triangolo
input int             InpWidth=2;                // Spessore delle linee del triangolo
input bool            InpFill=false;             // Riempie il triangolo con colore
input bool            InpBack=false;             // Sottofondo triangolo
input bool            InpSelection=true;         // Evidenzia movimento
input bool            InpHidden=true;            // Nascosto nella lista oggetti
input long            InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il triangolo dalle coordinate fornite                                   |
//+------------------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // ID del chart
                    const string          name="Triangle",   // nome del triangolo
                    const int             sub_window=0,      // indice sottofinestra
                    datetime              time1=0,           // orario del primo punto
                    double                price1=0,          // prezzo del primo punto
                    datetime              time2=0,           // orario del secondo pun
                    double                price2=0,          // prezzo del secondo pun
                    datetime              time3=0,           // orario del terzo punto
                    double                price3=0,          // prezzo del terzo punto
                    const color           clr=clrRed,        // colore del triangolo
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee del
                    const int             width=1,           // spessore delle linee d
                    const bool            fill=false,        // riempie il triangolo d
                    const bool            back=false,        // in sottofondo
                    const bool            selection=true,    // evidenzia movimento
                    const bool            hidden=true,       // nascosta nella lista d
                    const long            z_order=0)         // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il triangolo dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,ti
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il triangolo! Error code = ",GetLastError());
      return(false);
     }
//--- imposta il colore del triangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee del triangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee del triangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- abilita (true) o disabilita (false) la modalità di riempimento del triangolo
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostra in primo piano (false) o sottofondo (true)
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di evidenziazione del triangolo
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta i punti di ancoraggio del triangolo                                   |
//+------------------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,      // ID del chart
                         const string name="Triangle", // nome del triangolo
                         const int    point_index=0, // indice del punto di ancoraggio
                         datetime     time=0,          // coordinate tempo, del punto di anco
                         double       price=0)           // coordinate di prezzo del p
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Delete the triangle                                                          |
//+------------------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,      // ID del chart
                    const string name="Triangle") // nome del triangolo
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il triangolo
```

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l'ellisse! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio del triangoloo ed imposta i             |
//| valori di default per quelli vuoti                                                 |
//+------------------------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, è posizionato 9 barre meno dalla
   if(!time2)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- imposta il secondo punto, 9 barre in meno dalla prima
      time2=temp[0];
     }
//--- se il prezzo del secondo punto non è impostato, lo sposta di 300 punti in meno
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se l'orario del terzo punto non è impostato, coincide con la data del secondo pu
   if(!time3)
      time3=time2;
//--- se il prezzo del terzo punto non è impostato, è uguale a quello del primo punto
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
```

```
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare i punti delle coordinate di ancoraggio del triangolo
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il triangolo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crea il triangolo
   if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio dell triangolo
```

```
//---contatore del ciclo
   int v_steps=accuracy*3/10;
//--- move the first anchor point
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p1>1)
         p1-=1;
      //--- sposta il punto
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int h_steps=bars*9/20-1;
//--- sposta il secondo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d2>1)
         d2-=1;
      //--- sposta il punto
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   v_steps=accuracy/4;
//--- sposta il terzo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p3<accuracy-1)
         p3+=1;
      //--- sposta il punto
```

```
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il triangolo dal grafico
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ELLIPSE

Ellisse.

Nota

Per ellisse, la modalità di riempimento con colore può essere impostata utilizzando la proprietà OBJPROP_FILL.

Esempio

Il seguente script crea e sposta l'ellisse sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script crea l'ellisse nel chart."
#property description "Vengono impostate le coordinate dei punti di ancoraggio"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpName="Ellisse";         // Nome dell'Ellisse
input int             InpDate1=30;               // data del 1mo punto, %
input int             InpPrice1=20;              // prezzo del 1mo punto, %
input int             InpDate2=70;               // data del 2ndo punto, %
input int             InpPrice2=80;              // prezzo del 2ndo punto, %
input int             InpDate3=50;               // data del 3zo punto, %
input int             InpPrice3=60;              // prezzo del 3zo punto, %
input color           InpColor=clrRed;           // Colore dell'ellisse
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stile delle linee dell'ellisse
input int             InpWidth=2;                // Spessore delle linee dell'ellisse
input bool            InpFill=false;             // Riempie l' ellisse con colore
input bool            InpBack=false;             // Sottofondo ellisse
input bool            InpSelection=true;         // Evidenzia movimento
input bool            InpHidden=true;            // Nascosto nella lista oggetti
input long            InpZOrder=0;               // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea un'ellisse dalle coordinate date                                        |
//+------------------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // ID del chart
                   const string          name="Ellipse",    // nome dell'ellisse
                   const int             sub_window=0,      // indice sottofinestra
                   datetime              time1=0,           // orario del primo punto
                   double                price1=0,          // prezzo del primo punto
                   datetime              time2=0,           // orario del secondo punt
                   double                price2=0,          // prezzo del secondo punt
                   datetime              time3=0,           // orario del terzo punto
                   double                price3=0,          // prezzo del terzo punto
                   const color           clr=clrRed,        // colore dell'ellisse
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // stile delle linee dell
                   const int             width=1,           // spessore delle linee de
                   const bool            fill=false,        // riempie l'ellisse con c
                   const bool            back=false,        // in sottofondo
                   const bool            selection=true,    // evidenzia movimento
                   const bool            hidden=true,       // nascosto nella lista og
                   const long            z_order=0)         // priorità per il click c
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- resetta il valore dell' errore
   ResetLastError();
//--- Crea un'ellisse dalle coordinate date
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l'ellisse! Error code = ",GetLastError());
      return(false);
     }
//--- imposta il colore dell'ellisse
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile delle linee dell'ellisse
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore delle linee dell'ellisse
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- abilita (true) o disabilita (false) la modalità di riempimento dell'ellisse
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

```
//--- abilita (true) o disabilita (false) la modalità di evidenziazione dell'ellisse c
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio dell'ellisse                                   |
//+------------------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // ID del chart
                        const string name="Ellipse", // nome dell'ellisse
                        const int    point_index=0, // indice del punto di ancoraggio
                datetime     time=0,          // coordinate del punto di ancoraggio te
                       double       price=0)        // coordinate prezzo, del punto di
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina ellisse                                                              |
//+------------------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // ID del chart
                   const string name="Ellipse") // nome dell'ellisse
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina un'ellisse
   if(!ObjectDelete(chart_ID,name))
```

```
      {
       Print(__FUNCTION__,
             ": fallimento nell'eliminare un ellisse! Error code = ",GetLastError());
       return(false);
      }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla i valori dei punti di ancoraggio dell'ellisse ed imposta i valori di dei
//| per quelli vuoti                                                             |
//+------------------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- se l'orario del primo punto non è impostato, sarà sulla barra corrente
   if(!time1)
      time1=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se l'orario del secondo punto non è impostato, è posizionato 9 barre meno dalla
   if(!time2)
     {
      //--- array per la ricezione dell'orario di apertura delle ultime 10 barre
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- imposta il secondo punto, 9 barre in meno dalla prima
      time2=temp[0];
     }
//--- se il prezzo del secondo punto non è impostato, lo sposta di 300 punti in meno r
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se l'orario del terzo punto non è impostato, coincide con la data del secondo pu
   if(!time3)
      time3=time2;
//--- se il prezzo del terzo punto non è impostato, è uguale a quello del primo punto
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
```

```
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate dei punti di ancoraggio dell'ellisse
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare l'ellisse
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- crea un ellisse
   if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta i punti di ancoraggio dell'ellisse
//---contatore del ciclo
```

```
   int v_steps=accuracy/5;
//--- sposta il primo ed il secondo punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa i seguenti valori
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- slitta i punti
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int h_steps=bars/5;
//--- sposta il terzo punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d3>1)
         d3-=1;
      //--- sposta il punto
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'ellisse dal grafico
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
```

```
}
```

OBJ_ARROW_THUMB_UP

Segno Pollice Su.

Nota

La posizione del Punto di ancoraggio relativa al segno può essere scelta tra l'enumerazione ENUM_ARROW_ANCHOR.

Segni larghi (più di 5) possono essere creati solo impostando l'appropriato valore della proprietà OBJPROP_WIDTH quando si scrive un codice in MetaEditor.

Esempio

Il seguente script crea e muove il segno di Pollice Su sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna il segno di \"Pollice Su\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpName="ThumbUp";     // Nome del segno
input int                InpDate=75;            // data del punto di ancoraggio, in %
input int                InpPrice=25;           // Anchor point price in %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;  // Tipo di ancoraggio
input color              InpColor=clrRed;       // Colore del segno
```

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Stile del bordo della linea
input int               InpWidth=5;           // Grandezza del segno
input bool              InpBack=false;        // Segno di sottofondo
input bool              InpSelection=true;    // evidenza spostamento
input bool              InpHidden=true;       // Nascosta nella lista oggetti
input long              InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il segno di Pollice Su                                                  |
//+------------------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long              chart_ID=0,           // ID del chart
                        const string            name="ArrowUp",       // nome del segno
                        const int               sub_window=0,         // indice sottofines
                              datetime          time=0,               // punto di anco
                        double                  price=0,              // prezzo punto di ar
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo di ancorago
                        const color             clr=clrRed,           // colore del segno
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // stile del bordo
                        const int               width=3,              // grandezza del se
                        const bool              back=false,           // in sottofondo
                        const bool              selection=true,       // evidenzia del r
                        const bool              hidden=true,          // nascosto nella
                        const long              z_order=0)            // priorità per il
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Pollice Su\"! Error code = ",Get
      return(false);
     }
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,        // ID del chart
                      const string name="ThumbUp",    // nome oggetto
                      datetime     time=0,            // coordinate tempo, del punto di
                      double       price=0)           // coordinate prezzo, del punto d
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il tipo di segno di ancoraggio Pollice Su                             |
//+------------------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // ID del cha
                              const string            name="ThumbUp",    // nome ogget
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo di an
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il tipo di ancora
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di ancora! Error code = ",GetLastError
      return(false);
     }
```

```
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il segno di Pollice Su                                                     |
//+------------------------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,       // ID del chart
                        const string name="ThumbUp") // nome del segno
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del segno \"Pollice Su\"! Error code = ",G
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default           |
//| per quelli vuoti                                                                   |
//+------------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
```

```
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate del segno di ancoraggio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il segno
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crea il segno di Pollice Su sul chart
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio e modificare la sua posizione rispetto al seg
//---contatore del ciclo
   int h_steps=bars/4;
//--- sposta il punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d>1)
         d-=1;
      //--- sposta il punto
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
```

```
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int v_steps=accuracy/4;
//--- sposta il punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p<accuracy-1)
         p+=1;
      //--- sposta il punto
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- cambia la locazione del punto di ancoraggio relativa al segno
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redisegna il chart
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il segno dal chart
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_THUMB_DOWN

Segno Pollice Giu.

Nota

La posizione del Punto di ancoraggio relativa al segno può essere scelta tra l'enumerazione ENUM_ARROW_ANCHOR.

Segni larghi (più di 5) possono essere creati solo impostando l'appropriato valore della proprietà OBJPROP_WIDTH quando si scrive un codice in MetaEditor.

Esempio

Il seguente script crea e muove il segno di Pollice Giù sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna il segno di \"Pollice Giu\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in percent
#property description " della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpName="ThumbDown";      // nome del segno
input int                InpDate=25;               // Data del punto di ancoraggio in %
input int                InpPrice=75;              // Punto di ancoraggio "prezzo", in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // tipo di ancoraggio
```

```
input color             InpColor=clrRed;         // Colore del segno
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Stile bordo linea
input int               InpWidth=5;              // Grandezza del segno
input bool              InpBack=false;           // Segno di Sottofondo
input bool              InpSelection=true;       // evidenza spostamento
input bool              InpHidden=true;          // Nascosta nella lista oggetti
input long              InpZOrder=0;             // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il segno di Pollice Giu                                                 |
//+------------------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // ID del chai
                          const string            name="ThumbDown",     // nome del se
                          const int               sub_window=0,         // indice sott
                          datetime                time=0,               // orario punt
                          double                  price=0,              // prezzo punt
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo di anc
                          const color             clr=clrRed,           // colore del
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // stile del l
                          const int               width=3,              // grandezza d
                          const bool              back=false,           // in sottofor
                          const bool              selection=true,       // evidenzia d
                          const bool              hidden=true,          // nascosto ne
                          const long              z_order=0)            // priorità pe
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Pollice Giu\"! Error code = ",GetLastError());
      return(false);
     }
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                      |
//+------------------------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,        // ID del chart
                        const string name="ThumbDown", // nome oggetto
                        datetime     time=0,           // coordinate orarie del punto
                        double       price=0)          // coordinate di prezzo del pun
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia il tipo di segno di ancoraggio Pollice Giu                                  |
//+------------------------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long              chart_ID=0,        // ID del c
                                const string            name="ThumbDown",  // nome ogg
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo di
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il tipo di ancora
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di ancora! Error code = ",GetLastError
      return(false);
```

```
      }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina il segno di Pollice Giu                                  |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // ID del chart
                          const string name="ThumbDown", // nome del segno
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del segno \"Pollice Giu\"! Error code = ",
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default |
//| per quelli vuoti                                                 |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
```

```
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate del segno di ancoraggio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il segno
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crea il segno di Pollice Giu sul chart
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio e modificare la sua posizione rispetto al seg
//---contatore del ciclo
   int h_steps=bars/4;
//--- sposta il punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d<bars-1)
         d+=1;
      //--- sposta il punto
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
```

```
         return;
       //--- ridisegna il chart
       ChartRedraw();
       // 0.05 secondi di ritardo
       Sleep(50);
      }
//--- 1 secondo di ritardo
   Sleep(1000);
//---contatore del ciclo
   int v_steps=accuracy/4;
//--- sposta il punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p>1)
         p-=1;
      //--- sposta il punto
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- cambia la locazione del punto di ancoraggio relativa al segno
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redisegna il chart
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il segno dal chart
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_UP

Segno Freccia Su.

Nota

La posizione del Punto di ancoraggio relativa al segno può essere scelta tra l'enumerazione ENUM_ARROW_ANCHOR.

Segni larghi (più di 5) possono essere creati solo impostando l'appropriato valore della proprietà OBJPROP_WIDTH quando si scrive un codice in MetaEditor.

Esempio

Il seguente script crea e muove il segno di Freccia Su sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna il segno di \"Freccia Su\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpName="ArrowUp"; // Nome del segno
input int                InpDate=25;           // Anchor point date in %
input int                InpPrice=25;          // Anchor point price in %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP; // Tipo di ancoraggio
input color              InpColor=clrRed;      // Colore del segno
```

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Stile del bordo della linea
input int               InpWidth=5;           // Grandezza del segno
input bool              InpBack=false;        // Segno di sottofondo
input bool              InpSelection=false;   // Evidenzia di movimento
input bool              InpHidden=true;       // Nascosta nella lista oggetti
input long              InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il segno di Freccia Su                                                  |
//+------------------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // ID del chart
                   const string            name="ArrowUp",       // nome del segno
                   const int               sub_window=0,         // indice sottofinest
                   datetime                time=0,               // punto di ancoraggi
                   double                  price=0,              // prezzo punto di ar
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo di ancoraggio
                   const color             clr=clrRed,           // colore del segno
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // stile del bordo de
                   const int               width=3,              // grandezza del segr
                   const bool              back=false,           // in sottofondo
                   const bool              selection=true,       // evidenzia del movi
                   const bool              hidden=true,          // nascosto nella lis
                   const long              z_order=0)            // priorità per il cl
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Freccia Su\"! Error code = ",Get
      return(false);
     }
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,       // ID del chart
                 const string name="ArrowUp",   // nome oggetto
                 datetime     time=0,           // coordinate del punto di ancoraggio t
                 double       price=0)          // anchor point price coordinate
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il tipo di segno di ancoraggio Freccia Giu                            |
//+------------------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // ID del chart
                         const string            name="ArrowUp",    // nome oggetto
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo di ancorag
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la posizione del punto di ancoraggio
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di ancora! Error code = ",GetLastError
      return(false);
     }
```

```
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il segno di Freccia Su                                                     |
//+------------------------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // ID del chart
                   const string name="ArrowUp") // nome del segno
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del segno \"Freccia Su\"! Error code = ",G
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default           |
//| per quelli vuoti                                                                   |
//+------------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
```

```
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate del segno di ancoraggio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il segno
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crea il segno di Freccia Su sul chart
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio e modificare la sua posizione rispetto al seg
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- sposta il punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p<accuracy-1)
         p+=1;
      //--- sposta il punto
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
```

```
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- cambia la locazione del punto di ancoraggio relativa al segno
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redisegna il chart
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il segno dal chart
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_DOWN

Segno Freccia Giù

Nota

La posizione del Punto di ancoraggio relativa al segno può essere scelta tra l'enumerazione ENUM_ARROW_ANCHOR.

Segni larghi (più di 5) possono essere creati solo impostando l'appropriato valore della proprietà OBJPROP_WIDTH quando si scrive un codice in MetaEditor.

Esempio

Il seguente script crea e muove il segno di Freccia Giù sul grafico. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna il segno di \"Freccia Giù\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpName="ArrowDown";      // nome del segno
input int                InpDate=75;               // Punto di ancoraggio "data", in %
input int                InpPrice=75;              // Punto di ancoraggio "prezzo", in %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // Tipo di ancora
input color              InpColor=clrRed;          // Colore del segno
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;       // Stile bordo linea
```

```
input int                InpWidth=5;              // Grandezza del segno
input bool               InpBack=false;           // Segno di Sottofondo
input bool               InpSelection=false;      // Evidenzia di movimento
input bool               InpHidden=true;          // Nascosta nella lista oggetti
input long               InpZOrder=0;             // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il segno di Freccia Giù                                                 |
//+------------------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,           // ID del chart
                     const string            name="ArrowDown",     // nome del segno
                     const int               sub_window=0,         // indice sottofine
                     datetime                time=0,               // orario punto di
                     double                  price=0,              // prezzo punto di
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo di ancorag
                     const color             clr=clrRed,           // colore del segn
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // stile del bordo
                     const int               width=3,              // grandezza del se
                     const bool              back=false,           // in sottofondo
                     const bool              selection=true,       // evidenzia per mu
                     const bool              hidden=true,          // nascosto nella l
                     const long              z_order=0)            // priorità per il
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Freccia Giu\"! Error code = ",Ge
      return(false);
     }
//--- tipo di ancoraggio
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

```
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli oc
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                      |
//+------------------------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,        // ID del chart
                   const string name="ArrowDown", // nome dell'oggetto
                   datetime     time=0,           // coordinate orarie del punto di ar
                   double       price=0)          // coordinate di prezzo del punto di
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia il tipo di segno di ancoraggio Freccia Giu                                  |
//+------------------------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,        // ID del chart
                           const string            name="ArrowDown",  // nome dell'ogg
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo di ancor
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la posizione del punto di ancoraggio
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di ancora! Error code = ",GetLastError
      return(false);
     }
//--- esecuzione avvenuta
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina il segno di Freccia Giù                                  |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,       // ID del chart
                     const string name="ArrowDown") // nome del segno
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del segno \"Freccia Giu\"! Error code = ",
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default |
//| per quelli vuoti                                                 |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
```

```
//--- per impostare e cambiare le coordinate del segno di ancoraggio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il segno
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crea il segno di Freccia Giu sul chart
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio e modificare la sua posizione rispetto al seg
//---contatore del ciclo
   int v_steps=accuracy/2;
//--- sposta il punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p>1)
         p-=1;
      //--- sposta il punto
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
```

```
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- cambia la locazione del punto di ancoraggio relativa al segno
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redisegna il chart
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il segno dal chart
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_STOP

Segno di Stop.

Nota

La posizione del Punto di ancoraggio relativa al segno può essere scelta tra l'enumerazione ENUM_ARROW_ANCHOR.

Segni larghi (più di 5) possono essere creati solo impostando l'appropriato valore della proprietà OBJPROP_WIDTH quando si scrive un codice in MetaEditor.

Esempio

Il seguente script crea e muove il segno di Stop sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna il segno di \"Stop\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpName="ArrowStop";      // Nome del segno
input int                InpDate=10;               // Data de punto di ancoraggio in %
input int                InpPrice=50;              // prezzo del punto di ancoraggio, i
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // tipo di ancoraggio
input color              InpColor=clrRed;          // Colore del segno
```

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Stile bordo linea
input int               InpWidth=5;              // Grandezza del segno
input bool              InpBack=false;           // Segno di Sottofondo
input bool              InpSelection=false;      // Evidenzia di movimento
input bool              InpHidden=true;          // Nascosta nella lista oggetti
input long              InpZOrder=0;             // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il segno di Stop                                                        |
//+------------------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,           // ID del chart
                     const string            name="ArrowStop",     // nome del segno
                     const int               sub_window=0,         // indice sottofine
                     datetime                time=0,               // orario punto di
                     double                  price=0,              // prezzo punto di
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo di ancorag
                     const color             clr=clrRed,           // colore del segn
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // stile del bordo
                     const int               width=3,              // grandezza del s
                     const bool              back=false,           // in sottofondo
                     const bool              selection=true,       // evidenzia per m
                     const bool              hidden=true,          // nascosto nella
                     const long              z_order=0)            // priorità per il
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Stop\" ! Error code = ",GetLastE
      return(false);
     }
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione de
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,       // ID del chart
                   const string name="ArrowStop", // nome dell'oggetto
                   datetime     time=0,           // coordinate orarie del punto di ar
                   double       price=0)          // coordinate di prezzo del punto di
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il tipo di segno di ancoraggio Stop                                   |
//+------------------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,        // ID del chart
                           const string            name="ArrowStop",  // nome dell'ogg
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // posizione pun
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il tipo di ancora
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di ancora! Error code = ",GetLastError
      return(false);
     }
```

```
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il segno di Stop                                                           |
//+------------------------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // ID del chart
                     const string name="ArrowStop") // nome etichetta
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del segno \"Stop\"! Error code = ",GetLast
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default           |
//| per quelli vuoti                                                                   |
//+------------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
```

```
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate del segno di ancoraggio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il segno
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creare il segno Stop sul chart
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio e modificare la sua posizione rispetto al seg
//---contatore del ciclo
   int h_steps=bars*2/5;
//--- sposta il punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d<bars-1)
         d+=1;
      //--- sposta il punto
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
```

```
      //--- ridisegna il chart
      ChartRedraw();
      // 0.025 secondi di ritardo
      Sleep(25);
     }
//--- cambia la locazione del punto di ancoraggio relativa al segno
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- redisegna il chart
   ChartRedraw();
//---contatore del ciclo
   h_steps=bars*2/5;
//--- sposta il punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d<bars-1)
         d+=1;
      //--- sposta il punto
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.025 secondi di ritardo
      Sleep(25);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il segno dal chart
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_CHECK

Segno di Controllo.

Nota

La posizione del Punto di ancoraggio relativa al segno può essere scelta tra l'enumerazione ENUM_ARROW_ANCHOR.

Segni larghi (più di 5) possono essere creati solo impostando l'appropriato valore della proprietà OBJPROP_WIDTH quando si scrive un codice in MetaEditor.

Esempio

Il seguente script crea e muove il segno di Check sul grafico. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna il segno \"Check\"."
#property description "Le coordinate del punto di ancoraggio sono impostate in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpName="ArrowCheck"; // Nome del segno
input int                InpDate=10;           // data del punto di ancoraggio, in %
input int                InpPrice=50;          // prezzo del punto di ancoraggio, in %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP; // Tipo di ancoraggio
input color              InpColor=clrRed;      // Colore del segno
```

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Stile del bordo della linea
input int               InpWidth=5;           // Grandezza del segno
input bool              InpBack=false;        // Segno di sottofondo
input bool              InpSelection=false;   // Evidenzia di movimento
input bool              InpHidden=true;       // Nascosta nella lista oggetti
input long              InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il segno Check                                                          |
//+------------------------------------------------------------------------------+
bool ArrowCheckCreate(const long              chart_ID=0,           // ID del chart
                      const string            name="ArrowCheck",    // nome del segno
                      const int               sub_window=0,         // indice sottofi
                      datetime                time=0,               // punto di ancora
                      double                  price=0,              // punto di ancora
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo di ancorag
                      const color             clr=clrRed,           // colore del segn
                      const ENUM_LINE_STYLE   style=STYLE_SOLID,    // stile bordo lin
                      const int               width=3,              // grandezza segno
                      const bool              back=false,           // in sottofondo
                      const bool              selection=true,       // evidenzia del m
                      const bool              hidden=true,          // nascosto nella
                      const long              z_order=0)            // priorità per il
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Check\" ! Error code = ",GetLast
      return(false);
     }
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,        // ID del chart
                    const string name="ArrowCheck", // nome oggetto
                    datetime     time=0,            // coordinate tempo, del punto di
                    double       price=0)           // coordinate prezzo, del punto di
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il tipo di ancoraggio Check                                           |
//+------------------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // ID del chart
                            const string            name="ArrowCheck", // nome oggetto
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo di anco
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il tipo di ancora
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di ancora! Error code = ",GetLastError
      return(false);
     }
```

```
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il segno di Check                                                          |
//+------------------------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // ID del chart
                      const string name="ArrowCheck") // nome del segno
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il segno \"Check\"! Error code = ",GetLastErr
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default           |
//| per quelli vuoti                                                                   |
//+------------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
```

```
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate del segno di ancoraggio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare il segno
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- creare il segno Check sul chart
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio e modificare la sua posizione rispetto al seg
//---contatore del ciclo
   int h_steps=bars*2/5;
//--- sposta il punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d<bars-1)
         d+=1;
      //--- sposta il punto
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
```

```
      //--- ridisegna il chart
      ChartRedraw();
      // 0.025 secondi di ritardo
      Sleep(25);
     }
//--- cambia la locazione del punto di ancoraggio relativa al segno
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redisegna il chart
   ChartRedraw();
//---contatore del ciclo
   h_steps=bars*2/5;
//--- sposta il punto di ancoraggio
   for(int i=0;i<h_steps;i++)
     {
      //--- usa il seguente valore
      if(d<bars-1)
         d+=1;
      //--- sposta il punto
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.025 secondi di ritardo
      Sleep(25);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il segno dal chart
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_LEFT_PRICE

Etichetta Prezzo a Sinistra

Esempio

Il seguente script crea e sposta l'etichetta prezzo sinistra, sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script crea l'etichetta prezzo sinistra sul chart."
#property description "Le coordinate del punto di ancoraggio sono impostate in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string            InpName="LeftPrice";  // Nome dell'etichetta prezzo
input int               InpDate=100;          // data del punto di ancoraggio in %
input int               InpPrice=10;          // prezzo del punto di ancoraggio in %
input color             InpColor=clrRed;      // Colore dell'etichetta del prezzo
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // stile del bordo della linea
input int               InpWidth=2;           // grandezza dell'etichetta prezzo
input bool              InpBack=false;        // sottofondo etichetta
input bool              InpSelection=true;    // evidenza spostamento
input bool              InpHidden=true;       // Nascosta nella lista oggetti
input long              InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea l'etichetta prezzo a sinistra                                           |
```

```
//+------------------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // ID del chart
                          const string          name="LeftPrice",  // nome dell'etiche
                          const int             sub_window=0,      // indice sottofine
                          datetime              time=0,            // orario del punto
                          double                price=0,           // prezzo del punto
                          const color           clr=clrRed,        // colore dell'etic
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // stile del bordo
                          const int             width=1,           // grandezza dell'e
                          const bool            back=false,        // in sottofondo
                          const bool            selection=true,    // evidenza movimen
                          const bool            hidden=true,       // nascosto nella l
                          const long            z_order=0)         // priorità per il
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea l'etichetta prezzo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione dell'etichetta prezzo a sinistra! Error code
      return(false);
     }
//--- imposta il colore dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
```

```
bool ArrowLeftPriceMove(const long   chart_ID=0,        // ID del chart
                        const string name="LeftPrice", // nome etichetta
                        datetime     time=0,           // coordinate orarie del punto
                        double       price=0)          // coordinate di prezzo del pun
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Delete the left price label from the chart                                   |
//+------------------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,       // ID del chart
                          const string name="LeftPrice") // nome etichetta
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'etichetta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione dell'etichetta prezzo a sinistra! Error co
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default     |
//| per quelli vuoti                                                             |
//+------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
```

```
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e modificare le coordinate del punto di ancoraggio etichetta
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare l'etichetta
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crea l'etichetta prezzo a sinistra sul chart
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

```
      }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio
//---contatore del ciclo
   int v_steps=accuracy*4/5;
//--- sposta il punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p<accuracy-1)
         p+=1;
      //--- sposta il punto
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'etichetta dal chart
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_RIGHT_PRICE

Etichetta, prezzo destra.

Esempio

Il seguente script crea e sposta l' etichetta prezzo a destra, sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script crea l'etichetta prezzo destra sul chart."
#property description "Le coordinate del punto di ancoraggio sono impostate in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string            InpName="RightPrice"; // Nome dell'etichetta prezzo
input int               InpDate=0;            // data del punto di ancoraggio, in %
input int               InpPrice=90;          // prezzo del punto di ancoraggio in %
input color             InpColor=clrRed;      // Colore dell'etichetta del prezzo
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // stile del bordo della linea
input int               InpWidth=2;           // grandezza dell'etichetta prezzo
input bool              InpBack=false;        // sottofondo etichetta
input bool              InpSelection=true;    // evidenza spostamento
input bool              InpHidden=true;       // Nascosta nella lista oggetti
input long              InpZOrder=0;          // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea l'etichetta prezzo a destra                                             |
```

```
//+------------------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // ID del chart
                           const string          name="RightPrice", // nome dell'etich
                           const int             sub_window=0,      // indice sottofi
                           datetime              time=0,            // orario punto di
                           double                price=0,           // prezzo del punt
                           const color           clr=clrRed,        // colore dell'eti
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // stile del bordo
                           const int             width=1,           // grandezza dell
                           const bool            back=false,        // in sottofondo
                           const bool            selection=true,    // evidenza movime
                           const bool            hidden=true,       // nascosto nella
                           const long            z_order=0)         // priorità per il
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea l'etichetta prezzo
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione dell'etichetta prezzo a destra! Error code =
      return(false);
     }
//--- imposta il colore dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
```

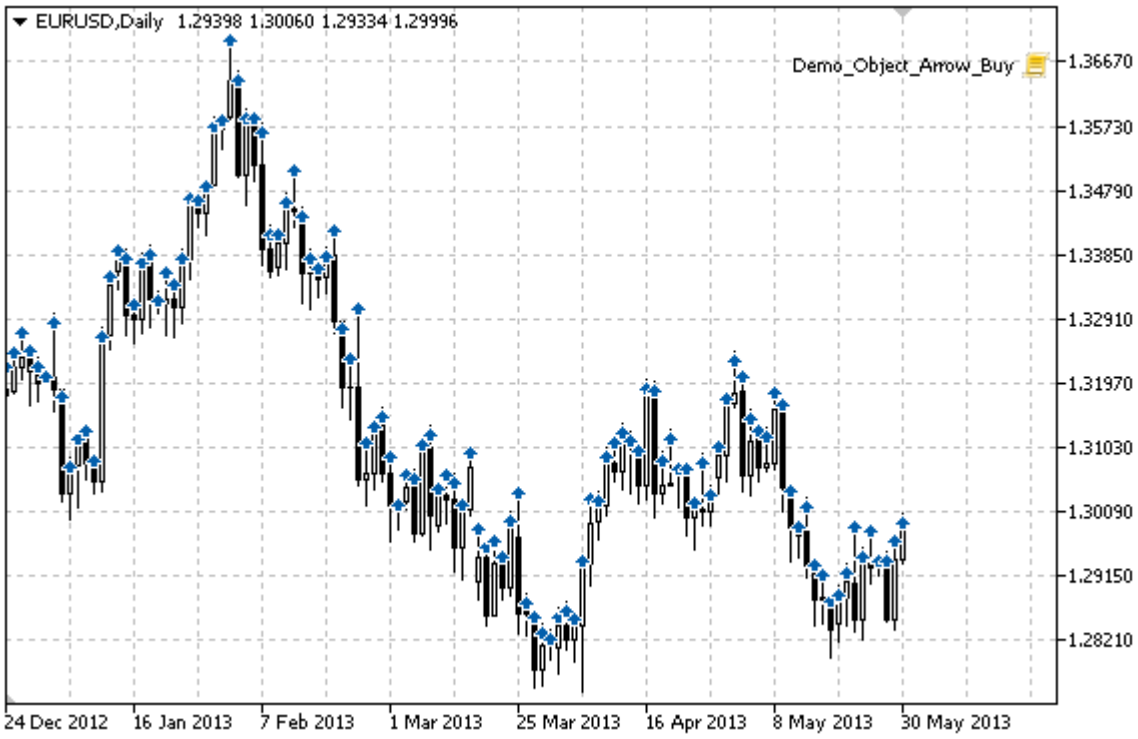
```
bool ArrowRightPriceMove(const long   chart_ID=0,        // ID del chart
                         const string name="RightPrice", // nome etichetta
                         datetime     time=0,            // coordinate temporali del p
                         double       price=0)           // coordinate di prezzo del p
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina l'etichetta prezzo a destra dal chart                                |
//+------------------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // ID del chart
                           const string name="RightPrice") // nome etichetta
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'etichetta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione dell'etichetta prezzo a destra! Error code
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default     |
//| per quelli vuoti                                                             |
//+------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
```

```
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e modificare le coordinate del punto di ancoraggio etichetta
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare l'etichetta
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- crea l'etichetta prezzo a destra sul chart
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

```
      }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta il punto di ancoraggio
//---contatore del ciclo
   int v_steps=accuracy*4/5;
//--- sposta il punto di ancoraggio
   for(int i=0;i<v_steps;i++)
     {
      //--- usa il seguente valore
      if(p>1)
         p-=1;
      //--- sposta il punto
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'etichetta dal chart
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_ARROW_BUY

Segno Buy.



Esempio

Il seguente script crea e muove il segno Buy sul grafico. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script disegna segni \"Buy\" nella finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input color InpColor=C'3,95,172'; // Colore dei segni
//+------------------------------------------------------------------------------+
//| Crea il segno Buy                                                            |
//+------------------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // ID del chart
                    const string          name="ArrowBuy",   // nome del segno
                    const int             sub_window=0,      // indice sottofinestra
                    datetime              time=0,            // orario del punto di ar
                    double                price=0,           // prezzo del punto di ar
                    const color           clr=C'3,95,172',   // colore del segno
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // stile linea (quando ev
                    const int             width=1,           // spessore linea (quando
                    const bool            back=false,        // in sottofondo
                    const bool            selection=false,   // evidenziazione di spos
```

```
                    const bool            hidden=true,       // nascosta nella lista d
                    const long            z_order=0)         // priorità per il click
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Buy\"! Error code = ",GetLastErr
      return(false);
     }
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea (quando evidenziata)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza della linea (quando evidenziata)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // ID del chart
                  const string name="ArrowBuy", // object name
                  datetime     time=0,          // coordinate tempo, del punto di anco
                  double       price=0)         // coordinate prezzo, del punto di anc
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Elimina il segno di Buy                                          |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,       // ID del chart
                    const string name="ArrowBuy") // nome del segno
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del segno \"Buy\"! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default       |
//| per quelli vuoti                                                 |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
   datetime date[]; // array per memorizzare le date di barre visibili
   double   low[];  // array per memorizzare i prezzi Low delle barre visibili
   double   high[]; // array memorizzare i prezzi High di barre visibili
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

```
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- riempie l'array dei prezzi Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Fallimento nel copiare i valori dei prezzi Low! Error code = ",GetLastEr
      return;
     }
//--- riempie l'array di prezzi High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Fallimento nel copiare i valori dei prezzi High! Error code = ",GetLastE
      return;
     }
//--- creare i segni Buy nel punto Low per ogni barra visibile
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- sposta i segni Buy signs al punto High per ogni barra visibile
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
```

```
//--- elimina i segni Buy
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//---
  }
```

OBJ_ARROW_SELL

Segno di Sell.

Esempio

Il seguente script crea e sposta il segno di Sell sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Script draws \"Sell\" segni nella finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input color InpColor=C'225,68,29'; // Colore dei segni
//+------------------------------------------------------------------------------+
//| Crea il segno Sell                                                           |
//+------------------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // ID del chart
                     const string          name="ArrowSell",  // nome del segno
                     const int             sub_window=0,      // indice sottofinestra
                     datetime              time=0,            // orario punto di ancor
                     double                price=0,           // prezzo punto di ancor
                     const color           clr=C'225,68,29',  // colore del segno
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // stile linea (quando e
                     const int             width=1,           // spessore linea (quanc
                     const bool            back=false,        // in sottofondo
                     const bool            selection=false,   // evidenzia per muovere
```

```
                     const bool            hidden=true,       // nascosto nella lista
                     const long            z_order=0)         // priorità per il clic
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il segno
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nella creazione del segno \"Sell\"! Error code = ",GetLastEr
      return(false);
     }
//--- imposta il colore del segno
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea (quando evidenziata)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza della linea (quando evidenziata)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare il segno con il mous
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                    |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,       // ID del chart
                   const string name="ArrowSell", // nome dell'oggetto
                   datetime     time=0,           // coordinate orarie del punto di an
                   double       price=0)          // coordinate di prezzo del punto di
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
      {
       Print(__FUNCTION__,
             ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
       return(false);
      }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il segno Sell                                                              |
//+------------------------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,       // ID del chart
                     const string name="ArrowSell") // nome del segno
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il segno
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del segno \"Sell\"! Error code = ",GetLast
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default           |
//| per quelli vuoti                                                                   |
//+------------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
   datetime date[]; // array per memorizzare le date di barre visibili
   double   low[];  // array per memorizzare i prezzi Low delle barre visibili
   double   high[]; // array memorizzare i prezzi High di barre visibili
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

```
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- riempie l'array dei prezzi Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Fallimento nel copiare i valori dei prezzi Low! Error code = ",GetLastEr
      return;
     }
//--- riempie l'array di prezzi High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Fallimento nel copiare i valori dei prezzi High! Error code = ",GetLastE
      return;
     }
//--- crea il segno Sell nel punto High per ogni barra visibile
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- sposta il segno Sell nel punto Low per ogni barra visibile
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
```

```
//--- elimina il segno Sell
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//---
  }
```

OBJ_ARROW

Oggetto Freccia.

Nota

La posizione del punto di ancoraggio relativa all'oggetto può essere selezionata da ENUM_ARROW_ANCHOR.

Frecce grandi (più di 5) possono essere create solo impostando l'appropriato valore della proprietà OBJPROP_WIDTH quando si scrive un codice MetaEditor.

Il tipo freccia necessaria può essere selezionata impostando uno dei codici simbolo del carattere Wingdings.

Esempio

Lo script che segue crea un oggetto Arrow sul grafico e cambia il suo tipo. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script crea una freccia casuale nella finestra chart."
#property description "Le coordinate del punto di ancoraggio sono impostate in"
#property description "percentuale della grandezza della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string            InpName="Arrow";        // Nome della freccia
input int               InpDate=50;             // data del punto di ancoraggio in %
```

```
input int                InpPrice=50;             // prezzo del punto di ancoraggio, in
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;    // Tipo di ancoraggio
input color              InpColor=clrDodgerBlue;  // Colore della Freccia
ingressoENUM_LINE_STYLE  InpStyle=STYLE_SOLID;    // Confine stile di linea
input int                InpWidth=10;             // Grandezza della Freccia
input bool               InpBack=false;           // Sottofondo Freccia
input bool               InpSelection=false;      // Evidenzia di movimento
input bool               InpHidden=true;          // Nascosta nella lista oggetti
input long               InpZOrder=0;             // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea la freccia |
//+------------------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // ID del chart
                 const string            name="Arrow",         // nome della freccia
                 const int               sub_window=0,         // indice sottofinestra
                 datetime                time=0,               // punto di ancoraggio
                 double                  price=0,              // prezzo punto di anco
                 const uchar             arrow_code=252,       // codice freccia
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // posizione punto di a
                 const color             clr=clrRed,           // colore freccia
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // stile del bordo del
                 const int               width=3,              // grandezza della fre
                 const bool              back=false,           // in sottofondo
                 const bool              selection=true,       // evidenzia del movime
                 const bool              hidden=true,          // nascosto nella lista
                 const long              z_order=0)            // priorità per il clic
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeArrowEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- creare una freccia
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare la freccia! Error code = ",GetLastError());
      return(false);
     }
//--- imposta il codice freccia
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore della freccia
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo linea
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza della freccia
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostare la freccia con il mo
//--- quando si crea un oggetto grafico utilizzando la funzione ObjectCreate, l'oggett
//--- evidenziato e mosso, per default. All'interno di questo metodo, la selezione dei
//--- è true per default, il che consente di evidenziare e spostare l'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                |
//+------------------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // ID del chart
               const string name="Arrow", // nome dell'oggetto
               datetime     time=0,       // coordinate orario, del punto di ancoraggi
               double       price=0)      // coordinate prezzo, del punto di ancoraggi
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Modifica il codice freccia                                                   |
//+------------------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // ID del chart
                     const string name="Arrow", // nome dell'oggetto
                     const uchar  code=252)     // codice freccia
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il codice freccia
```

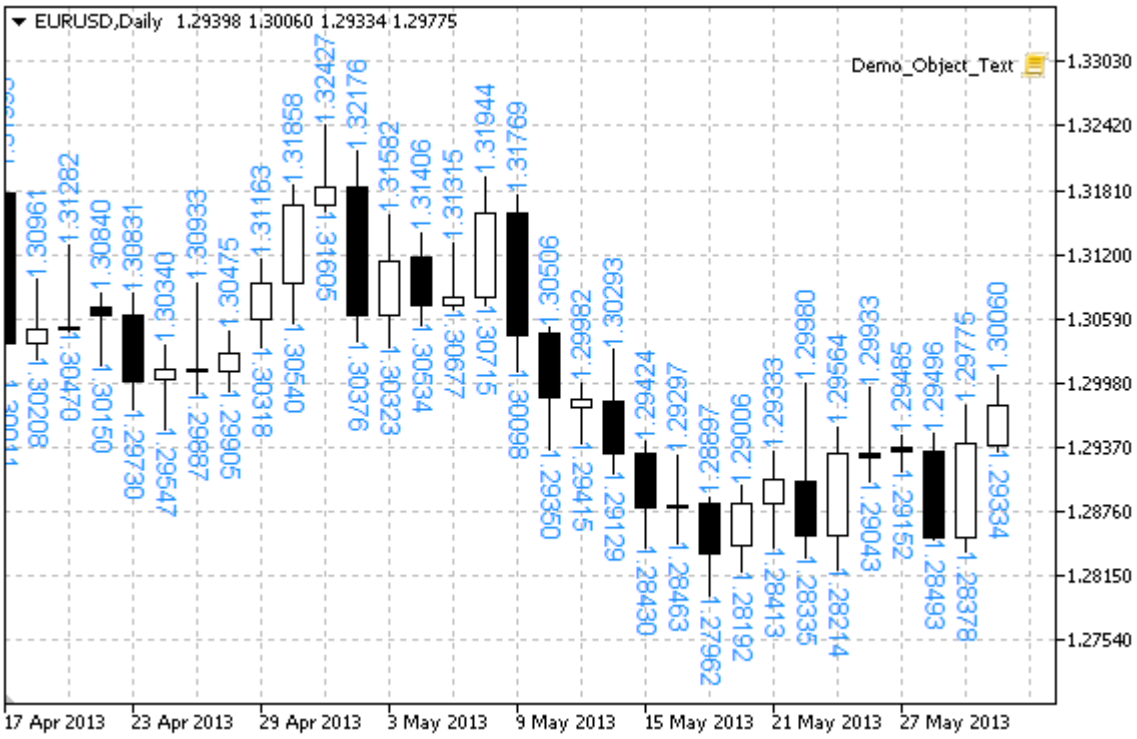
```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il codice freccia! Error code = ",GetLastError
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia il tipo di ancoraggio                                                       |
//+------------------------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // ID del chart
               const string name="Arrow", // nome dell'oggetto
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo di ancoraggio
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il tipo di ancora
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di ancora! Error code = ",GetLastError
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//--- creare una freccia                                                               |
//+------------------------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // ID del chart
                 const string name="Arrow") // nome della freccia
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina la freccia
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare la freccia! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default           |
//| per quelli vuoti                                                                   |
```

```
//+------------------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- grandezza dell'array prezzo
   int accuracy=1000;
//--- array per la memorizzazione dei valori di data e prezzo da essere usati
//--- per impostare e cambiare le coordinate del segno di ancoraggio
   datetime date[];
   double   price[];
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
/--- riempie l'array dei prezzi
//--- trova i valori più alti e più bassi del chart
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definisce un cambio di step del prezzo e riempie l'array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definisce i punti per disegnare la freccia
   int d=InpDate*(bars-1)/100;
```

```
   int p=InpPrice*(accuracy-1)/100;
//--- crea la freccia sul grafico
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart
   ChartRedraw();
//--- considera tutti i casi di creazione delle frecce in loop
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // ritardo di mezzo secondo
      Sleep(500);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina la freccia dal grafico
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_TEXT

Oggetto Testo



Nota

La posizione del punto di ancoraggio relativa al testo può essere scelta dall'enumerazione ENUM_ANCHOR_POINT. È inoltre possibile modificare la pendenza del testo con la proprietàOBJPROP_ANGLE.

Esempio

Il seguente script crea diversi oggetti Testo sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script crea l'oggetto grafico \"Testo\"."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpFont="Arial";         // Font
input int                InpFontSize=10;          // Grandezza del font
input color              InpColor=clrRed;         // Colore
input double             InpAngle=90.0;           // Angolo di pendenza in gradi
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;    // Tipo di ancora
input bool               InpBack=false;           // Oggetto di sottofondo
input bool               InpSelection=false;      // Evidenzia di movimento
input bool               InpHidden=true;          // Nascosta nella lista oggetti
```

```
input long              InpZOrder=0;             // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea l'oggetto Testo                                                         |
//+------------------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,               // ID del chart
                const string            name="Text",              // nome dell'oggetto
                const int               sub_window=0,             // indice sottofines
                datetime                time=0,                   // orario punto di a
                double                  price=0,                  // prezzo punto di a
                const string            text="Text",              // il testo stesso
                const string            font="Arial",             // font
                const int               font_size=10,             // grandezza del for
                const color             clr=clrRed,               // colore
                const double            angle=0.0,                // inclinazione del
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // tipo di ancora
                const bool              back=false,               // in sottofondo
                const bool              selection=false,          // evidenzia movimen
                const bool              hidden=true,              // nascosto nella li
                const long              z_order=0)                // priorità per il o
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeTextEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea l'oggetto Testo
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l'oggetto \"Testo\"! Error code = ",GetLastError
      return(false);
     }
//--- imposta il testo
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- imposta il font
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- imposta grandezza font
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- imposta l'angolo di inclinazione del testo
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di spostamento dell'oggetto con
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli o
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta il punto di ancoraggio                                                      |
//+------------------------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // ID del chart
              const string name="Text", // nome dell'oggetto
              datetime     time=0,      // coordinate tempo, del punto di ancoraggio
              double       price=0)     // coordinate prezzo del punto di ancoraggio
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia il testo dell'oggetto                                                       |
//+------------------------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // ID del chart
                const string name="Text", // nome dell'oggetto
                const string text="Text") // testo
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il testo dell'oggetto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il testo! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Elimina l'oggetto Testo                                          |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // ID del chart
                const string name="Text") // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l'oggetto \"Testo\"! Error code = ",GetLastEr
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default       |
//| per quelli vuoti                                                 |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
   datetime date[]; // array per memorizzare le date di barre visibili
   double   low[];  // array per memorizzare i prezzi Low delle barre visibili
   double   high[]; // array memorizzare i prezzi High di barre visibili
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

```
      {
       Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
       return;
      }
//--- riempie l'array dei prezzi Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
      {
       Print("Fallimento nel copiare i valori dei prezzi Low! Error code = ",GetLastEr
       return;
      }
//--- riempie l'array di prezzi High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
      {
       Print("Fallimento nel copiare i valori dei prezzi High! Error code = ",GetLastE
       return;
      }
//--- definisce quanto spesso il testo viene visualizzato
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- definisce lo step
   int step=1;
   switch(scale)
      {
       case 0:
          step=12;
          break;
       case 1:
          step=6;
          break;
       case 2:
          step=4;
          break;
       case 3:
          step=2;
          break;
      }
//--- crea il testo per i valori High e Low delle barre (con i gaps)
   for(int i=0;i<bars;i+=step)
      {
       //--- crea il testo
       if(!TextCreate(0,"TextHigh_"+(string)i,0,date[i],high[i],DoubleToString(high[i],
          InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
       if(!TextCreate(0,"TextLow_"+(string)i,0,date[i],low[i],DoubleToString(low[i],5),
          InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
```

```
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- elimina il testo
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//---
  }
```

OBJ_LABEL

Oggetto Etichetta.

Nota

La posizione del punto di ancoraggio relativa all'etichetta può essere scelta dall'enumerazione ENUM_ANCHOR_POINT. Le coordinate del punto di ancoraggio sono fissate in pixel.

È anche possibile selezionare l'angolo di ancoraggio dell'etichetta testo dall'enumerazione ENUM_BASE_CORNER.

Esempio

Il seguente script crea e sposta l'oggetto Modifica sul grafico. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script crea l'oggetto grafico \"Etichetta\"."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string            InpName="Label";          // Nome Etichetta
input int               InpX=150;                 // distanza asse X
input int               InpY=150;                 // distanza asse Y
input string            InpFont="Arial";          // Font
input int               InpFontSize=14;           // Grandezza del Font
input color             InpColor=clrRed;          // Colore
input double            InpAngle=0.0;             // Angolo di inclinazione in gradi
```

```
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Tipo di ancoraggio
input bool              InpBack=false;           // Oggetto di sottofondo
input bool              InpSelection=true;       // evidenza spostamento
input bool              InpHidden=true;          // Nascosta nella lista oggetti
input long              InpZOrder=0;             // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea un'etichetta di testo                                                   |
//+------------------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // ID del chart
                 const string            name="Label",             // nome dell'etiche
                 const int               sub_window=0,             // indice sottofine
                 const int               x=0,                      // coordinate X
                 const int               y=0,                      // coordinate Y
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // angolo del chart
                 const string            text="Label",             // testo
                 const string            font="Arial",             // font
                 const int               font_size=10,             // grandezza font
                 const color             clr=clrRed,               // colore
                 const double            angle=0.0,                // pendenza testo
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // tipo di ancorag
                 const bool              back=false,               // in sottofondo
                 const bool              selection=false,          // evidenzia movier
                 const bool              hidden=true,              // nascosto nella
                 const long              z_order=0)                // priorità per il
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea un'etichetta di testo
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare un'etichetta di testo! Error code = ",GetLastErr
      return(false);
     }
//--- imposta le coordinate dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- imposta l'angolo del chart, relativo a quali punti coordinate vengono definiti
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- imposta il testo
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- imposta il font
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- imposta grandezza font
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- imposta l'angolo di inclinazione del testo
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
```

```
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta l'etichetta di testo                                                  |
//+------------------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // ID del chart
               const string name="Label", // nome dell'etichetta
               const int    x=0,          // coordinate X
               const int    y=0)          // coordinate Y
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta l'etichetta di testo
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento delle coordinate X dell'etichetta! Error c
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento della coordinata Y dell'etichetta! Error c
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia angolo del chart per collegare l'etichetta                            |
//+------------------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,               // ID del char
                       const string            name="Label",            // nome dell'e
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // angolo del
  {
//--- resetta il valore dell' errore
   ResetLastError();
```

```
//--- cambia l'angolo di ancoraggio
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'angolo di ancoraggio! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia l'etichetta testo                                                           |
//+------------------------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // ID del chart
                     const string name="Chart", // nome dell'oggetto
                     const string text="Text")  // testo
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il testo dell'oggetto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il testo! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina l'etichetta di testo                                                       |
//+------------------------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // ID del chart
                 const string name="Label", // nome dell'etichetta
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'etichetta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l'etichetta di testo! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
```

```
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- memorizza le coordinate dell'etichetta nelle variabili locali
   int x=InpX;
   int y=InpY;
//--- grandezza della finestra chart
   long x_distance;
   long y_distance;
//--- imposta la grandezza della finestra
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Fallimento nell'ottenere la grandezza del chart! Error code = ",GetLastEr
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Fallimento nell'ottenere l'altezza del chart! Error code = ",GetLastError
      return;
     }
//--- imposta la correttezza dei parametri di input
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- prepara il testo iniziale per l'etichetta
   string text;
   StringConcatenate(text,"Upper left corner: ",x,",",y);
//--- crea un'etichetta di testo sul chart
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per mezzo secondo
   ChartRedraw();
   Sleep(500);
//--- sposta l'etichetta e modifica il testo, simultaneamente
//--- numero di iterazioni per assi
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- sposta l'etichetta verso il basso
   for(int i=0;i<v_steps;i++)
     {
      //--- cambia le coordinate
      y+=2;
      //--- sposta l'etichetta e cambia il suo testo
      MoveAndTextChange(x,y,"Angolo sinistro superiore: ");
```

```
      }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- sposta l'etichetta sulla destra
   for(int i=0;i<h_steps;i++)
     {
      //--- cambia le coordinate
      x+=2;
      //--- sposta l'etichetta e cambia il suo testo
      MoveAndTextChange(x,y,"Angolo sinistro superiore: ");
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- sposta l'etichetta verso l'alto
   for(int i=0;i<v_steps;i++)
     {
      //--- cambia le coordinate
      y-=2;
      //--- sposta l'etichetta e cambia il suo testo
      MoveAndTextChange(x,y,"Angolo sinistro superiore: ");
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- sposta l'etichetta a sinistra
   for(int i=0;i<h_steps;i++)
     {
      //--- cambia le coordinate
      x-=2;
      //--- sposta l'etichetta e cambia il suo testo
      MoveAndTextChange(x,y,"Angolo sinistro superiore: ");
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- ora, sposta il punto cambiando l'angolo di ancoraggio
//--- spostandolo verso l'angolo inferiore sinistro
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- cambia il testo dell'etichetta
   StringConcatenate(text,"Lower left corner: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redraw the chart and wait for two seconds
   ChartRedraw();
   Sleep(2000);
//--- spostato all'angolo inferiore destro
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- cambia il testo dell'etichetta
   StringConcatenate(text,"Lower right corner: ",x,",",y);
```

```
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redraw the chart and wait for two seconds
   ChartRedraw();
   Sleep(2000);
//--- spostato all'angolo superiore destro
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- cambia il testo dell'etichetta
   StringConcatenate(text,"Upper right corner: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redraw the chart and wait for two seconds
   ChartRedraw();
   Sleep(2000);
//--- spostandolo all'angolo superiore sinistro
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- cambia il testo dell'etichetta
   StringConcatenate(text,"Upper left corner: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redraw the chart and wait for two seconds
   ChartRedraw();
   Sleep(2000);
//--- elimina l'etichetta
   LabelDelete(0,InpName);
//--- redisegna il chart ed attende per mezzo secondo
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------------------+
//| La funzione sposta l'oggetto e cambia il suo testo                           |
//+------------------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- sposta l'etichetta
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- cambia il testo dell'etichetta
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- verifica se il funzionamento dello script è stato forzatamente disattivato
   if(IsStopped())
      return(false);
//--- redisegna il chart
   ChartRedraw();
```

```
// 0.01 secondi di ritardo
   Sleep(10);
//--- esce dalla funzione
   return(true);
  }
```

OBJ_BUTTON

Oggetto bottone.

Nota

Le coordinate del punto di ancoraggio sono impostate in pixels. È possibile selezionare l'angolo di ancoraggio del bottone da ENUM_BASE_CORNER.

Esempio

The following script creates and moves Button object on the chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script crea il bottone sul chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string           InpName="Button";            // Nome bottone
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Angolo del chart per l'ancorag
input string           InpFont="Arial";             // Font
input int              InpFontSize=14;              // Grandezza Font
input color            InpColor=clrBlack;           // Colore del testo
input color            InpBackColor=C'236,233,216'; // Colore sottofondo
input color            InpBorderColor=clrNONE;      // Colore bordo
input bool             InpState=false;              // Premuto/Rilasciato
input bool             InpBack=false;               // Oggetto sottofondo
input bool             InpSelection=false;          // Evidenzia movimento
```

```
input bool              InpHidden=true;              // Nascosto nella lista oggetti
input long              InpZOrder=0;                 // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea il bottone                                                              |
//+------------------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // ID del chart
                  const string            name="Button",            // nome del bottor
                  const int               sub_window=0,             // indice sottofir
                  const int               x=0,                      // coordinate X
                  const int               y=0,                      // coordinate Y
                  const int               width=50,                 // spessore bottor
                  const int               height=18,                // altezza bottone
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // angolo del char
                  const string            text="Button",            // testo
                  const string            font="Arial",             // font
                  const int               font_size=10,             // grandezza font
                  const color             clr=clrBlack,             // colore del test
                  const color             back_clr=C'236,233,216',  // coloredi sottof
                  const color             border_clr=clrNONE,       // colore del bord
                  const bool              state=false,              // premuto/rilasci
                  const bool              back=false,               // in sottofondo
                  const bool              selection=false,          // evidenzia movie
                  const bool              hidden=true,              // nascosto nella
                  const long              z_order=0)                // priorità per il
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il bottone
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare il bottone! Error code = ",GetLastError());
      return(false);
     }
//--- imposta coordinate bottone
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- imposta grandezza bottone
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- imposta l'angolo del chart, relativo a quali punti coordinate vengono definiti
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- imposta il testo
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- imposta il font
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- imposta grandezza font
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- set text color
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta colore di background
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- imposta colore del bordo
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- set button state
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- abilita (true) o disabilita (false) la modalità di movimento del bottone con il
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta il bottone                                                            |
//+------------------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,    // ID del chart
                const string name="Button", // nome del bottone
                const int    x=0,           // coordinate X
                const int    y=0)           // coordinate Y
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il bottone
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento delle coordinate X del bottone! Error code
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento delle coordinate Y del bottone! Error code
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia la grandezza del bottone                                              |
//+------------------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,    // ID del chart
```

```
                      const string name="Button", // nome del bottone
                      const int    width=50,      // spesso del bottone
                      const int    height=18)     // altezza del bottone
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la grandezza del bottone
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare lo spessore del bottone! Error code = ",GetLast
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'altezza del bottone! Error code = ",GetLastEr
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia l'angolo del chart per collegare il bottone                           |
//+------------------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,               // ID del cha
                        const string           name="Button",            // nome del b
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // angolo del
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia l'angolo di ancoraggio
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'angolo di ancoraggio! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il testo del bottone                                                  |
//+------------------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,    // ID del chart
                      const string name="Button", // nome del bottone
                      const string text="Text")   // testo
  {
//--- resetta il valore dell' errore
```

```
   ResetLastError();
//--- cambia il testo dell'oggetto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il testo! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il bottone                                                                 |
//+------------------------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // ID del chart
                  const string name="Button") // nome del bottone
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il bottone
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare il bottone! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- grandezza della finestra chart
   long x_distance;
   long y_distance;
//--- imposta la grandezza della finestra
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Fallimento nell'ottenere la grandezza del chart! Error code = ",GetLastEr
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Fallimento nell'ottenere l'altezza del chart! Error code = ",GetLastError
      return;
     }
//--- definisce lo step per il cambio della grandezza del bottone
```

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- imposta le coordinate del bottone e la sua grandezza
   int x=(int)x_distance/32;
   int y=(int)y_distance/32;
   int x_size=(int)x_distance*15/16;
   int y_size=(int)y_distance*15/16;
//--- crea il bottone
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSiz
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- redisegna il chart
   ChartRedraw();
//--- riduce il bottone nel loop
   int i=0;
   while(i<13)
     {
      //--- ritardo di mezzo secondo
      Sleep(500);
      //--- commuta il bottone allo stato premuto
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- ridisegna il chart ed aspetta 0.2 secondi
      ChartRedraw();
      Sleep(200);
      //--- ridefinisce le coordinate e la grandezza del bottone
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //--- riduce il bottone
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- riporta indietro lo stato del bottone a rilasciato
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //--- ridisegna il chart
      ChartRedraw();
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- incrementa il contatore del loop
      i++;
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- elimina il bottone
   ButtonDelete(0,InpName);
   ChartRedraw();
```

```
//--- aspetta per 1 secondo
   Sleep(1000);
//---
  }
```

OBJ_CHART

Oggetto Chart.

Nota

Le coordinate del punto di ancoraggio sono impostate in pixels. Puoi selezionare l'angolo di ancoraggio dall'enumerazione ENUM_BASE_CORNER.

Simbolo, periodo e la scala possono essere selezionati per l'oggetto Chart. Modalità di visualizzazione della scala di prezzo e data possono anche essere attivate/disattivate.

Esempio

Lo script che segue crea e sposta l'ogetto Chart sul chart(grafico). Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Script creates \"Chart\" object."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string           InpName="Chart";              // Nome dell'oggetto
input string           InpSymbol="EURUSD";           // Simbolo
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;          // Periodo
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Angolo di ancoraggio
input int              InpScale=2;                   // Scala
input bool             InpDateScale=true;            // Mostra scala temporale
input bool             InpPriceScale=true;           // Mostra scala prezzo
```

```
input color            InpColor=clrRed;             // Colore del bordo quando eviden
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT;   // Stile della linea quando evide
input int              InpPointWidth=1;             // Grandezza punto da spostare
input bool             InpBack=false;               // Oggetto sottofondo
input bool             InpSelection=true;           // Evidanzia movimento
input bool             InpHidden=true;              // Nascosto nella lista oggetti
input long             InpZOrder=0;                 // Priorità per il click del mous
//+------------------------------------------------------------------------------+
//| Crea l'oggetto Chart                                                         |
//+------------------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,               // ID del ch
                       const string            name="Chart",             // nome ogge
                       const int               sub_window=0,             // indice so
                       const string            symbol="EURUSD",          // simoblo
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,         // periodo
                       const int               x=0,                      // coordinat
                       const int               y=0,                      // coordinat
                       const int               width=300,                // spessore
                       const int               height=200,               // altezza
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // angolo di
                       const int               scale=2,                  // scala
                       const bool              date_scale=true,          // mostra sc
                       const bool              price_scale=true,         // mostra sc
                       const color             clr=clrRed,               // colore de
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // stile del
                       const int               point_width=1,            // grandezza
                       const bool              back=false,               // in sottof
                       const bool              selection=false,          // evidenzia
                       const bool              hidden=true,              // nascosto
                       const long              z_order=0)                // priorità
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea l'oggetto Chart
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l'oggetto \"Chart\"! Error code = ",GetLastError
      return(false);
     }
//--- imposta le coordinate dell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- imposta la grandezza dell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- imposta l'angolo del chart, relativo a quali punti coordinate vengono definiti
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- imposta il simbolo
```

```
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- imposta il periodo
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- imposta la scala
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- mostra (true) o nascondi (false) la scala temporale
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- mostra (true) o nascondi (false) la scala prezzo
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- imposta il colore del bordo quando è abilitata la modalità di evidenziazione
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo della linea quando la modalità di evidenzia dell'ogge
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza di un punto di ancoraggio per lo spostamento di un oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta il simoboli ed il timeframe per l'oggetto del Chart                  |
//+------------------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,       // ID del
                                   const string          name="Chart",     // nome del
                                   const string          symbol="EURUSD",  // simbolo
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // time fra
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il simobolo e timeframe dell'oggetto Chart
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": fallimento nell'impostare un simbolo per l'oggetto \"Chart\"! Error coc
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
            ": fallimento nell'impostare un periodo per l'oggetto \"Chart\"! Error coc
      return(false);
```

```
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta l'oggetto Chart                                                       |
//+------------------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // ID del Chart (non quello dell'ogget
                     const string name="Chart", // nome dell'oggetto
                     const int    x=0,          // coordinate X
                     const int    y=0)          // coordinate Y
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta l'oggetto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare le coordinate X dell'oggetto \"Chart\"! Erro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare le coordinate Y dell'oggetto \"Chart\"! Erro
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia la grandezza dell'oggetto Chart                                       |
//+------------------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // ID del chart (non quello del
                           const string name="Chart", // nome dell'oggetto
                           const int    width=300,    // spessore
                           const int    height=200)   // altezza
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la grandezza dell'oggetto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare lo spessore dell'oggetto \"Chart\"! Error code
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
```

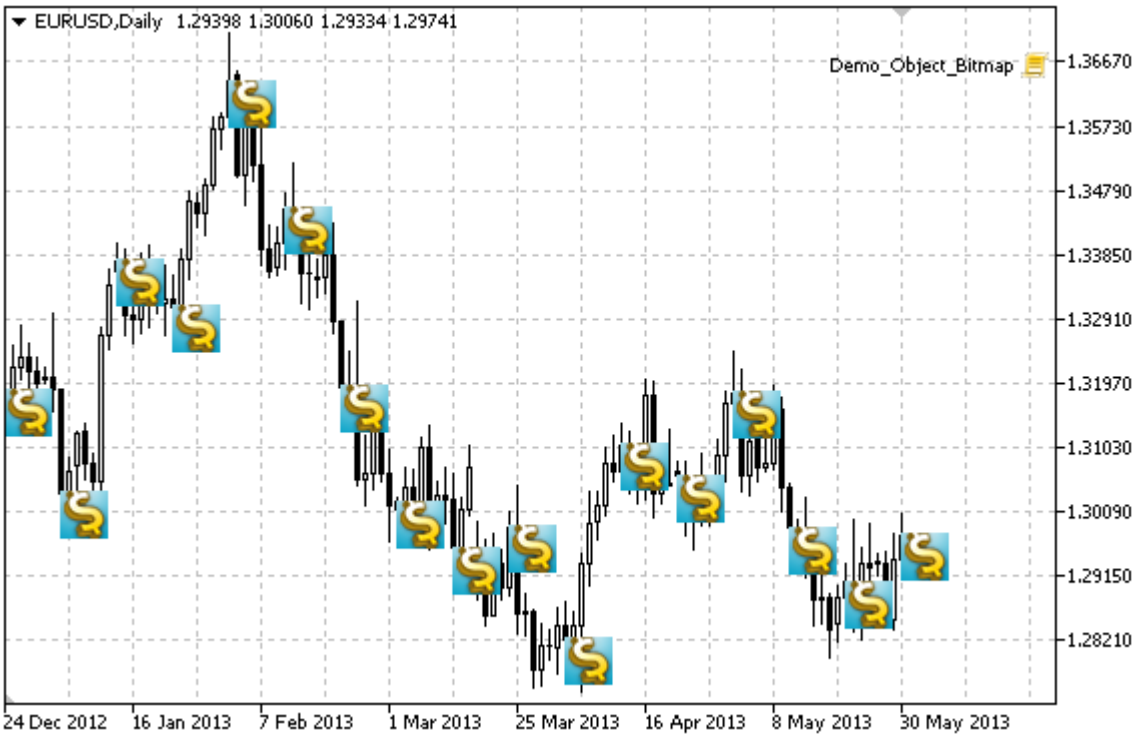
```
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'altezza dell'oggetto \"Chart\"! Error code =
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Restituisce l'ID dell'oggetto Chart                                          |
//+------------------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // ID del chart (non quello dell'ogge
                      const string name="Chart") // nome dell'oggetto
  {
//--- prepara la variabile per ottenere l'ID dell'oggetto Chart
   long id=-1;
//--- resetta il valore dell' errore
   ResetLastError();
//--- ottiene l' ID
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": fallimento nell'ottenere l'ID dell'oggetto \"Chart\"! Error code = ",Ge
     }
//--- restituisce il risultato
   return(id);
  }
//+------------------------------------------------------------------------------+
//| Elimina l'oggetto Chart                                                      |
//+------------------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // ID del chart (non quello dell'ogg
                       const string name="Chart") // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina il bottone
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l'oggetto \"Chart\"! Error code = ",GetLastEr
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
```

```
//--- ottiene il numero di simboli nel Market Watch
   int  symbols=SymbolsTotal(true);
//--- controlla se il simbolo con un nome specificato è presente nell'elenco dei simbo
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("Error! ",InpSymbol," il simbolo non è presente nel \"Market Watch\"!");
      return;
     }
//--- controlla la validità dei parametri di input
   if(InpScale<0 || InpScale>5)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }

//--- grandezza della finestra chart
   long x_distance;
   long y_distance;
//--- imposta la grandezza della finestra
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Fallimento nell'ottenere la grandezza del chart! Error code = ",GetLastEr
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Fallimento nell'ottenere l'altezza del chart! Error code = ",GetLastError
      return;
     }
//--- imposta le coordinate dell'oggetto Chart e la sua grandezza
   int x=(int)x_distance/16;
   int y=(int)y_distance/16;
   int x_size=(int)x_distance*7/16;
   int y_size=(int)y_distance*7/16;
//--- crea l'oggetto Chart
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,I
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
```

```
   Sleep(1000);
//--- distende l'oggetto Chart
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- resize
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart ed attende 0.01 secondi
      ChartRedraw();
      Sleep(10);
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- cambia il timeframe del chart
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- tre secondi di ritardo
   Sleep(3000);
//--- elimina l'oggetto
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- aspetta per 1 secondo
   Sleep(1000);
//---
  }
```

OBJ_BITMAP

Oggetto Bitmap



Nota

Per oggetti Bitmap, puoi selezionare l'ambito di visibilità di un'immagine.

Esempio

Lo script che segue crea diversi bitmaps sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Gli Script creano un bitmap nella finestra del chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string          InpFile="\\Images\\dollar.bmp"; // Nome del file del Bitmap
input int             InpWidth=24;                    // Ambito di visibilità, coordin
input int             InpHeight=24;                   // Ambito di visibilità, coordin
input int             InpXOffset=4;                   // Ambito di visibilità, slittar
input int             InpYOffset=4;                   // Ambito di visibilità, slittar
input color           InpColor=clrRed;                // Colore del bordo quando evide
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // Stile della linea quando vier
input int             InpPointWidth=1;                // Grandezza punto da muovere
input bool            InpBack=false;                  // Oggetto di sottofondo
input bool            InpSelection=false;             // Evidenzia di movimento
```

```
input bool            InpHidden=true;                  // Nascosto nella lista oggetti
input long            InpZOrder=0;                     // Priorità per il click del mou
//+------------------------------------------------------------------------------------+
//| Crea un bitmap nella finestra chart                                                |
//+------------------------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,        // ID del chart
                  const string          name="Bitmap",     // nome del bitmap
                  const int             sub_window=0,      // indice sottofinestra
                  datetime              time=0,            // orario punto di ancorag
                  double                price=0,           // prezzo punto di ancorag
                  const string          file="",           // nome del file bitmap
                  const int             width=10,          // ambito di visibilità co
                  const int             height=10,         // ambito di visibilità co
                  const int             x_offset=0,        // ambito di visibilità, s
                  const int             y_offset=0,        // ambito di visibilità, s
                  const color           clr=clrRed,        // colore del bordo quando
                  const ENUM_LINE_STYLE style=STYLE_SOLID, //stile linea quando evide
                  const int             point_width=1,     // grandezza del punto di
                  const bool            back=false,        // in sottofondo
                  const bool            selection=false,   // evidenzia movimento
                  const bool            hidden=true,       // nascosto nella lista og
                  const long            z_order=0)         // priorità per il click d
  {
//--- imposta coordinate punto di ancoraggio se non sono impostate
   ChangeBitmapEmptyPoint(time,price);
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea un bitmap
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare un bitmap nella finestra del chart! Error code =
      return(false);
     }
//--- imposta il percorso al file dell'immagine
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": fallimento nel caricare l'immagine! Error code = ",GetLastError());
      return(false);
     }
//--- imposta ambito di visibilità per l'immagine; se i valori dello spessore o dell'
//--- eccedono lo spessore e l'altezza(rispettivamente) di un'immagine sorgente,
//--- essa non viene disegnata; in caso contrario
//--- solo la parte corrispondente di questi valori viene disegnata
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- imposta la parte di un'immagine che dev'essere visualizzata nell'ambito visibil
//--- la parte di default è la parte superiore sinistra dell'immagine; i valori permet
```

```
//--- di eseguire uno slittamento da quest' area visualizzando un'altra parte dell'imm
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- imposta il colore del bordo quando è abilitata la modalità di evidenziazione
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo della linea quando la modalità di evidenzia dell'ogge
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza di un punto di ancoraggio per lo spostamento di un oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta una nuova immagine per il bitmap                                     |
//+------------------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,    // ID del chart
                    const string name="Bitmap", // nome del bitmap
                    const string file="")       // percorso del file
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il percorso al file dell'immagine
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": fallimento nel caricare l'immagine! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta un bitmap nella finestra del chart                                    |
//+------------------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,    // ID del chart
                const string name="Bitmap", // nome del bitmap
                datetime     time=0,        // orario del punto di ancoraggio
                double       price=0)       // prezzo del punto di ancoraggio
  {
//--- se il punto della posizione non è impostato, spostarlo nella barra corrente che
```

```
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta il punto di ancoraggio
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare il punto di ancoraggio! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia la grandezza dell'ambito visibilità (bitmap)                                |
//+------------------------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,     // ID del chart
                      const string name="Bitmap", // nome del bitmap
                      const int    width=0,       // spessore del bitmap
                      const int    height=0)      // altezza del bitmap
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la grandezza del bitmap
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare lo spessore del bitmap! Error code = ",GetLastE
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'altezza del bitmap! Error code = ",GetLastErr
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia le coordinate dell'angolo superiore sinistro dell'ambito di visibilità
//+------------------------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,     // ID del chart
                           const string name="Bitmap", // nome del bitmap
                           const int    x_offset=0,    // ambito di visibilità coordin
                           const int    y_offset=0)    // ambito di visibilità coordin
```

```
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia le coordinate dell'ambito di visibilità
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare le coordinate X dell'ambito di visibilità! Erro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare le coordinate Y dell'ambito di visibilità! Erro
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina il bitmap                                                                  |
//+------------------------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,     // ID del chart
                  const string name="Bitmap") // nome del bitmap
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'etichetta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione del bitmap! Error code = ",GetLastError())
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Controlla i valori di punto di ancoraggio ed imposta i valori di default           |
//| per quelli vuoti                                                                   |
//+------------------------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- se il tempo del punto non è impostato, sarà sulla barra corrente
   if(!time)
      time=TimeCurrent();
//--- se il prezzo del punto non è impostato, avrà un valore Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

```
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
   datetime date[];  // array per memorizzare le date delle barre visibili
   double   close[]; // array per memorizzare i prezzi Close
//--- nome del file bitmap
   string   file="\\Images\\dollar.bmp";
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- allocazione della memoria
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- riempie l'array dei prezzi Close
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("Fallimento nel copiare i valori dei prezzi Close! Error code = ",GetLastE
      return;
     }
//--- definisce quanto spesso devono essere visualizzate le immagini
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- definisce lo step
   int step=1;
   switch(scale)
     {
      case 0:
         step=27;
         break;
      case 1:
         step=14;
         break;
      case 2:
         step=7;
         break;
      case 3:
         step=4;
         break;
      case 4:
         step=2;
         break;
```

```
      }
//--- crea bitmap per valori High e Low delle barre (con i gap)
   for(int i=0;i<bars;i+=step)
     {
      //--- crea i bitmaps
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- elimina il segno Sell
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//---
  }
```

OBJ_BITMAP_LABEL

Oggetto Etichetta Bitmap.

Nota

La posizione del punto di ancoraggio relativa all'etichetta può essere scelta dall'enumerazione ENUM_ANCHOR_POINT. Le coordinate del punto di ancoraggio sono fissate in pixel.

Puoi anche selezionare l'angolo di ancoraggio bitmap dall'enumerazione ENUM_BASE_CORNER.

Per l'etichetta bitmap, puoi selezionare l'ambito di visibilità di un immagine.

Esempio

Lo script che segue crea diversi bitmaps sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script crea l'oggetto \"Bitmap Label\" ."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string            InpName="BmpLabel";              // nome dell'Etichetta
input string            InpFileOn="\\Images\\dollar.bmp"; // Nome del file per modalit
input string            InpFileOff="\\Images\\euro.bmp";  // Nome del File per modalit
input bool              InpState=false;                   // Etichetta premuta/rilasci
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;      // Angolo del chart per l'ar
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;          // Tipo di ancoraggio
input color             InpColor=clrRed;                  // Colore del bordo quando s
```

```
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;              // Stile della linea quando
input int                InpPointWidth=1;                   // Grandezza del punto da m
input bool               InpBack=false;                     // Oggetti di sottofondo
input bool               InpSelection=false;                // Evidenzia di movimento
input bool               InpHidden=true;                    // Nascosto nella lista ogg
input long               InpZOrder=0;                       // Priorità per il click del
//+------------------------------------------------------------------------------+
//| Crea l'oggetto Etichetta Bitmap                                              |
//+------------------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // ID del cha
                       const string            name="BmpLabel",          // nome dell
                       const int               sub_window=0,             // indice sot
                       const int               x=0,                      // coordinate
                       const int               y=0,                      // coordinate
                       const string            file_on="",               // immagine
                       const string            file_off="",              // immagine
                       const int               width=0,                  // ambito di
                       const int               height=0,                 // ambito di
                       const int               x_offset=10,              // ambito di
                       const int               y_offset=10,              // ambito di
                       const bool              state=false,              // premuto/ri
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // angolo del
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // tipo di an
                       const color             clr=clrRed,               // colore del
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // stile del
                       const int               point_width=1,            // grandezza
                       const bool              back=false,               // in sottofo
                       const bool              selection=false,          // evidenzia
                       const bool              hidden=true,              // nascosto
             const long              z_order=0)             // priorità per il click
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea un'etichetta bitmap
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l'oggetto \"Bitmap Label\"! Error code = ",GetLas
      return(false);
     }
//--- imposta l'immagine in modalità On e Off
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": fallimento nel caricare l'immagine per la modalità On! Error code = ",G
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
     {
```

```
      Print(__FUNCTION__,
            ": fallimento nel caricare l'immagine per la modalità Off! Error code = ",
      return(false);
     }
//--- imposta le coordinate dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- imposta ambito di visibilità per l'immagine; se i valori dello spessore o dell'a
//--- eccedono lo spessore e l'altezza(rispettivamente) di un'immagine sorgente,
//--- essa non viene disegnata; in caso contrario
//--- solo la parte corrispondente di questi valori viene disegnata
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- imposta la parte di un'immagine che dev'essere visualizzata nell'ambito visibili
//--- la parte di default è la parte superiore sinistra dell'immagine; i valori permet
//--- di eseguire uno slittamento da quest' area visualizzando un'altra parte dell'imm
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- definisce lo status dell'etichetta (premuta o rilasciata)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- imposta l'angolo del chart, relativo a quali punti coordinate vengono definiti
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- imposta il tipo di ancora
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- imposta il colore del bordo quando è abilitata la modalità di evidenziazione
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile del bordo della linea quando la modalità di evidenzia dell'ogge
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta la grandezza di un punto di ancoraggio per lo spostamento di un oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Imposta una nuova immagine per l'oggetto Etichetta Bitmap                    |
//+------------------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,      // ID del chart
                         const string name="BmpLabel", // nome dell'etichetta
                         const int    on_off=0,        // modificatore (On o Off)
                         const string file="")         // percorso del file
```

```
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- imposta il percorso al file dell'immagine
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": fallimento nel caricare l'immagine! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta l'oggetto Etichetta Bitmap                                            |
//+------------------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,      // ID del chart
                     const string name="BmpLabel", // nome etichetta
                     const int    x=0,             // coordinate X
                     const int    y=0)             // coordinate Y
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta l'oggetto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento delle coordinate X dell'oggetto! Error coo
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento della coordinata Y dell'oggetto! Error coo
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia grandezza ambito di visibilità (oggetto)                              |
//+------------------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,      // ID del chart
                           const string name="BmpLabel", // nome dell'etichetta
                           const int    width=0,         // spessore della linea
                           const int    height=0)        // altezza dell'etichetta
  {
//--- resetta il valore dell' errore
   ResetLastError();
```

```
//--- cambia la grandezza dell'oggetto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare lo spessore dell'oggetto! Error code = ",GetLas
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'altezza dell'oggetto! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------
//| Cambia le coordinate dell'angolo superiore sinistro dell'ambito di visibilità
//+------------------------------------------------------------------------------------
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // ID del chart
                                const string name="BmpLabel", // nome dell'etichetta
                                const int    x_offset=0,      // ambito di visibilità
                                const int    y_offset=0)      // ambito di visibilità
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia le coordinate dell'ambito di visibilità dell'oggetto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare le coordinate X dell'ambito di visibilità! Erro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare le coordinate Y dell'ambito di visibilità! Erro
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina l'oggetto "Etichetta Bitmap"                                         |
//+------------------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // ID del chart
                       const string name="BmpLabel") // nome dell'etichetta
  {
//--- resetta il valore dell' errore
```

```
   ResetLastError();
//--- elimina l'etichetta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminazione dell'oggetto \"Bitmap label\"! Error code
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- grandezza della finestra chart
   long x_distance;
   long y_distance;
//--- imposta la grandezza della finestra
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Fallimento nell'ottenere la grandezza del chart! Error code = ",GetLastEr
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Fallimento nell'ottenere l'altezza del chart! Error code = ",GetLastError
      return;
     }
//--- definisce le coordinate dell'etichetta bitmap
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- imposta la grandezza dell'etichetta e le coordinate dell'ambito di visibilità
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- imposta l'etichetta bitmap al centro della finestra
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidd
     {
      return;
     }
//--- ridisegna il chart e ne attende un secondo
   ChartRedraw();
   Sleep(1000);
//--- cambio dell'ambito di visibilità dell'etichetta nel loop
   for(int i=0;i<6;i++)
```

```
      {
       //--- cambia la grandezza dell'ambito di visibilità
       width--;
       height--;
       if(!BitmapLabelChangeSize(0,InpName,width,height))
          return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
          return;
       //--- ridisegna il chart
       ChartRedraw();
       // 0.3 secondi di ritardo
       Sleep(300);
      }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- cambia le coordinate dell'ambito di visibilità nel loop
   for(int i=0;i<2;i++)
     {
       //--- cambia le coordinate dell'ambito di visibilità
       x_offset++;
       y_offset++;
       if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
          return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
          return;
       //--- ridisegna il chart
       ChartRedraw();
       // 0.3 secondi di ritardo
       Sleep(300);
      }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina l'etichetta
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_EDIT

Oggetto Modifica.

Nota

Le coordinate del punto di ancoraggio sono impostate in pixels. Puoi selezionare l'angolo di ancoraggio Modifica dall'enumerazione ENUM_BASE_CORNER.

È anche possibile selezionare uno dei tipi di allineamento del testo all'interno di Modifica dall'enumerazione ENUM_ALIGN_MODE.

Esempio

Il seguente script crea e sposta l'oggetto Modifica sul grafico. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo Script crea l'oggetto \"Modifica\" ."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string            InpName="Edit";              // Nome dell'oggetto
input string            InpText="Text";              // Testo dell'oggetto
input string            InpFont="Arial";             // Font
input int               InpFontSize=14;              // Grandezza Font
input ENUM_ALIGN_MODE   InpAlign=ALIGN_CENTER;       // Tipo di allineamento testo
input bool              InpReadOnly=false;           // Permesso di modifica
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Angolo del chart per l'ancorag
input color             InpColor=clrBlack;           // Colore del testo
```

```
input color             InpBackColor=clrWhite;       // Colore di sottofondo
input color             InpBorderColor=clrBlack;     // Colore bordo
input bool              InpBack=false;               // Oggetto sottofondo
input bool              InpSelection=false;          // Evidenzia movimento
input bool              InpHidden=true;              // Nascosto nella lista oggetti
input long              InpZOrder=0;                 // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Crea Oggetto Modifica                                                        |
//+------------------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,               // ID del chart
                const string           name="Edit",              // nome dell'oggetto
                const int              sub_window=0,             // indice sottofinest
                const int              x=0,                      // Coordinate X
                const int              y=0,                      // Coordinate Y
                const int              width=50,                 // spessore
                const int              height=18,                // altezza
                const string           text="Text",              // testo
                const string           font="Arial",             // font
                const int              font_size=10,             // grandezza font
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,       // tipo di allineamer
                const bool             read_only=false,          // abilita per modifi
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // angolo del chart p
                const color            clr=clrBlack,             // colore del teato
                const color            back_clr=clrWhite,        // colore di sottofor
                const color            border_clr=clrNONE,       // colore del bordo
                const bool             back=false,               // in sottofondo
                const bool             selection=false,          // evidenzia moviment
                const bool             hidden=true,              // nascosto nella lis
                const long             z_order=0)                // priorità per il cl
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea il campo modifica
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l'oggetto \"Modifica\" ! Error code = ",GetLastEr
      return(false);
     }
//--- imposta le coordinate dell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- imposta la grandezza dell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- imposta il testo
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- imposta il font
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

```
//--- imposta grandezza font
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- imposta il tipo di allineamento di testo nell'oggetto
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- abilita (true) o annulla (false) la modalità sola-lettura
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- imposta l'angolo del chart, relativo a quali coordinate oggetto sono definite
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- set text color
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta colore di background
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- imposta colore del bordo
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta oggetto Modifica                                                            |
//+------------------------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,  // ID del chart
              const string name="Edit", // nome oggetto
              const int    x=0,         // Coordinate X
              const int    y=0)         // Coordinate Y
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta l'oggetto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento delle coordinate X dell'oggetto! Error coo
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento della coordinata Y dell'oggetto! Error coo
      return(false);
     }
```

```
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Ridimensione oggetto Modifica                                                |
//+------------------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // ID del chart
                    const string name="Edit", // nome oggetto
                    const int    width=0,     // spessore
                    const int    height=0)    // altezza
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la grandezza dell'oggetto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare lo spessore dell'oggetto! Error code = ",GetLas
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'altezza dell'oggetto! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il testo dell'oggetto Modifica                                        |
//+------------------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // ID del chart
                    const string name="Edit", // nome oggetto
                    const string text="Text") // testo
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il testo dell'oggetto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il testo! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
```

```
//| Restituisce il testo dell'oggetto Modifica                                      |
//+---------------------------------------------------------------------------------+
bool EditTextGet(string      &text,         // testo
                 const long   chart_ID=0,  // ID del chart
                 const string name="Edit") // nome oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- ottiene il testo dell'oggetto
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": fallimento nell'ottenere il testo! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+---------------------------------------------------------------------------------+
//| Elimina l'oggetto Modifica                                                      |
//+---------------------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // ID del chart
                const string name="Edit") // nome dell'oggetto
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'etichetta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l'oggetto \"Modifica\"! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+---------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                          |
//+---------------------------------------------------------------------------------+
voidOnStart()
  {
//--- grandezza della finestra chart
   long x_distance;
   long y_distance;
//--- imposta la grandezza della finestra
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Fallimento nell'ottenere la grandezza del chart! Error code = ",GetLastEr
      return;
```

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("Fallimento nell'ottenere l'altezza del chart! Error code = ",GetLastError
       return;
      }
//--- definisce lo step per cambiare il campo Modifica
   int x_step=(int)x_distance/64;
//--- imposta le coordinate del campo Modifica e la sua grandezza
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- memorizza il testo nella variabile locale
   string text=InpText;
//--- crea il campo modifica
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,
      InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,In
      {
       return;
      }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- allunga il campo Modifica
   while(x_size-x<x_distance*5/8)
      {
       //--- incrementa lo spessore del campo Modifica
       x_size+=x_step;
       if(!EditChangeSize(0,InpName,x_size,y_size))
          return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
          return;
       //--- ridisegna il grafico ed attende per 0.05 secondi
       ChartRedraw();
       Sleep(50);
      }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- cambia il testo
   for(int i=0;i<20;i++)
      {
       //--- aggiunge "+" all'inizio ed alla fine
       text="+"+text+"+";
       if(!EditTextChange(0,InpName,text))
          return;
       //--- controlla se l'operazione dello script è stata disabilitata per forza
       if(IsStopped())
```

```
         return;
      //--- redisegna il chart ed attende per 0.1 secondi
      ChartRedraw();
      Sleep(100);
     }
//--- ritardo di mezzo secondo
   Sleep(500);
//--- elimina il campo modifica
   EditDelete(0,InpName);
   ChartRedraw();
//--- aspetta per 1 secondo
   Sleep(1000);
//---
  }
```

OBJ_EVENT

Oggetto Evento.

Nota

Quando si passa il mouse sopra l'evento, viene visualizzato il testo.

Esempio

Lo script che segue crea e sposta l' oggetto Evento sul chart. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script disegna l'oggetto grafico \"Evento\"."
#property description "La data del punto di ancoraggio è impostata in percentuale dell
#property description "spessoore in barre della finestra chart."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string           InpName="Event";     // Nome evento
input int              InpDate=25;          // Data evento, %
input string           InpText="Text";      // Testo Evento
input color            InpColor=clrRed;     // Colore Evento
input int              InpWidth=1;          // Grandezza del punto quando evidenziato
input bool             InpBack=false;       // Sottofondo evento
input bool             InpSelection=false; // Evidenzia movimento
input bool             InpHidden=true;      // Nascosto nella lista oggetti
```

```
input long            InpZOrder=0;        // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//--- creare l'oggetto Evento sul chart |
//+------------------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // ID del chart
                 const string          name="Event",     // nome dell'evento
                 const int             sub_window=0,     // indice sottofinestra
                 const string          text="Text",      // testo dell'evento
                 datetime              time=0,           // orario
                 const color           clr=clrRed,       // colore
                 const int             width=1,          // spessore punto quando evide
                 const bool            back=false,       // in sottofondo
                 const bool            selection=false,  // evidenzia movimento
                 const bool            hidden=true,      // nascosto nella lista oggett
                 const long            z_order=0)        // priorità per il click del r
  {
//--- se l'orario non è impostato, crea l'oggetto sull'ultima barra
   if(!time)
      time=TimeCurrent();
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea l'oggetto Evento
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l'oggetto \"Evento\"! Error code = ",GetLastError
      return(false);
     }
//--- imposta il testo dell'evento
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- imposta il colore
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo spessore del punto di ancoraggio se l'oggetto è evidenziato
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) la modalità di movimento degli eventi da mou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il testo dell'oggetto Evento                                          |
//+------------------------------------------------------------------------------+
```

```
bool EventTextChange(const long   chart_ID=0,   // ID del chart
                     const string name="Event", // nome dell'evento
                     const string text="Text")  // testo
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il testo dell'oggetto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il testo! Error code = ",GetLastError());
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Sposta l'oggetto Evento                                                            |
//+------------------------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // ID del chart
               const string name="Event", // nome evento
               datetime     time=0)       // orario
  {
//--- se l'orario non è impostato, sposta l'evento sull'ultima barra
   if(!time)
      time=TimeCurrent();
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta l'oggetto
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostare l'oggetto \"Evento\"! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina l'oggetto Evento                                                           |
//+------------------------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // ID del chart
                 const string name="Event") // nome evento
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'oggetto
   if(!ObjectDelete(chart_ID,name))
     {
```

```
      Print(__FUNCTION__,
            ": fallimento nell'eliminare l'oggetto \"Evento\"! Error code = ",GetLastE
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta la correttezza dei parametri di input
   if(InpDate<0 || InpDate>100)
     {
      Print("Error! Valori non corretti dei parametri di input!");
      return;
     }
//--- Numero di barre visibili nella finestra del chart
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- array per memorizzare i valori data che devono essere usati
//--- per impostare e cambiare i punti di ancoraggio degli oggetti Evento
   datetime date[];
//--- allocazione della memoria
   ArrayResize(date,bars);
//--- riempie l'array delle date
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Fallimento nella copia dei valori tempo! Error code = ",GetLastError());
      return;
     }
//--- definisce i punti per creare un'oggetto
   int d=InpDate*(bars-1)/100;
//--- crea l'oggetto Evento
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- ora, sposta l'oggetto
//---contatore del ciclo
   int h_steps=bars/2;
//--- sposta l'oggetto
   for(int i=0;i<h_steps;i++)
     {
```

```
      //--- usa il seguente valore
      if(d<bars-1)
         d+=1;
      //--- sposta il punto
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart
      ChartRedraw();
      // 0.05 secondi di ritardo
      Sleep(50);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- elimina il canale dal chart
   EventDelete(0,InpName);
   ChartRedraw();
//--- 1 secondo di ritardo
   Sleep(1000);
//---
  }
```

OBJ_RECTANGLE_LABEL

Rectangle Label object.

Nota

Le coordinate del punto di ancoraggio sono impostate in pixels. Puoi selezionare l'angolo di ancoraggio dell'etichetta rettangolo dall'enumerazione ENUM_BASE_CORNER. Il tipo di bordo dell'etichetta rettangolo può essere selezionato dall'enumerazione ENUM_BORDER_TYPE.

L'oggetto viene utilizzato per creare e progettare l'interfaccia grafica personalizzata.

Esempio

Il seguente script crea e sposta l'etichetta Rettangolo sul grafico. Funzioni speciali sono state sviluppate per creare e modificare le proprietà dell'oggetto grafico. È possibile utilizzare queste funzioni "come è" nelle proprie applicazioni.

```
//--- descrizione
#property description "Lo script crea l'oggetto grafico \"Etichetta Rettangolo \"."
//--- mostra la finestra dei parametri di input durante il lancio dello script
#property script_show_inputs
//--- parametri di input dello script
input string             InpName="Label";          // Nome Etichetta
input color              InpBackColor=clrSkyBlue;        // Colore di sottofondo
input ENUM_BORDER_TYPE   InpBorder=BORDER_FLAT;        // Tipo di bordo
input ENUM_BASE_CORNER   InpCorner=CORNER_LEFT_UPPER; // Angolo del chart per l'ancorag
input color              InpColor=clrDarkBlue;         // Colore del bordo piatto (Flat)
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;         // Stile del bordo piatto (Flat)
input int                InpLineWidth=3;               // Spessore del bordo piatto (Flat
```

```
input bool              InpBack=false;              // Oggetto sottofondo
input bool              InpSelection=true;          // Evidanzia movimento
input bool              InpHidden=true;             // Nascosto nella lista oggetti
input long              InpZOrder=0;                // Priorità per il click del mouse
//+------------------------------------------------------------------------------+
//| Create rectangle label                                                       |
//+------------------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,               // ID del chart
                     const string           name="RectLabel",         // nome dell'et
                     const int              sub_window=0,             // indice sotto
                     const int              x=0,                      // coordinate X
                     const int              y=0,                      // coordinate Y
                     const int              width=50,                 // spessore
                     const int              height=18,                // altezza
                     const color            back_clr=C'236,233,216',  // colore di ba
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,     // tipo di bord
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // angolo del c
                     const color            clr=clrRed,               // colore del b
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,        // stile del bo
                     const int              line_width=1,             // spessore del
                     const bool             back=false,               // in sottofond
                     const bool             selection=false,          // evidenzia mo
                     const bool             hidden=true,              // nascosto nel
                     const long             z_order=0)                // priorità per
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- crea un' etichetta rettangolo
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": fallimento nel creare l'etichetta rettangolo! Error code = ",GetLastEr
      return(false);
     }
//--- imposta le coordinate dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- imposta la grandezza dell'etichetta
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- imposta colore di background
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- imposta il tipo di bordo
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- imposta l'angolo del chart, relativo a quali punti coordinate vengono definiti
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- imposta il colore del bordo piatto (in modalità Flat)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- imposta lo stile della linea del bordo piatto
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- imposta lo spessore del bordo piatto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- mostra in primo piano (false) o sottofondo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- abilita (true) o disabilita (false) il modo di spostamento dell'etichetta con il
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nascondi (true) o mostra (falso) il nome di oggetto grafico nella lista degli og
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- imposta la priorità per ricevere l'evento di un clic del mouse nel grafico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Sposta l'etichetta rettangolo                                                |
//+------------------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,       // ID del chart
                   const string name="RectLabel", // nome dell'etichetta
                   const int    x=0,              // coordinate X
                   const int    y=0)              // coordinate Y
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- sposta l'etichetta rettangolo
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento delle coordinate X dell'etichetta! Error c
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": fallimento nello spostamento della coordinata Y dell'etichetta! Error c
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Cambia lo spessore dell'etichetta rettangolo |
//+------------------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,       // ID del chart
                         const string name="RectLabel", // nome dell'etichetta
                                const int    width=50,         // spessore della linea
                                const int    height=18)        // altezza dell'etichetta
  {
```

```
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia la grandezza dell'etichetta
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare lo spessore dell'etichetta! Error code = ",GetL
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare l'altezza dell'etichetta! Error code = ",GetLas
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Cambia il tipo di bordo dell'etichetta rettangolo                                  |
//+------------------------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,           // ID del
                               const string           name="RectLabel",     // nome de
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // tipo di
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- cambia il tipo di bordo
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": fallimento nel cambiare il tipo di bordo! Error code = ",GetLastError()
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Elimina l'etichetta del rettangolo                                                 |
//+------------------------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,       // ID del chart
                     const string name="RectLabel") // nome etichetta
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- elimina l'etichetta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

```
            ": fallimento nell'eliminare l'etichetta del rettangolo! Error code = ",Ge
      return(false);
     }
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- grandezza della finestra chart
   long x_distance;
   long y_distance;
//--- imposta la grandezza della finestra
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Fallimento nell'ottenere la grandezza del chart! Error code = ",GetLastEi
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Fallimento nell'ottenere l'altezza del chart! Error code = ",GetLastError
      return;
     }
//--- definisce le coordinate dell'etichetta rettangolo
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- imposta la grandezza dell'etichetta
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- crea un' etichetta rettangolo
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- ridisegna il chart e ne attende un secondo
   ChartRedraw();
   Sleep(1000);
//--- cambia la grandezza dell'etichetta rettangolo
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- resize
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

```
      //--- controlla se l'operazione dello script è stata disabilitata per forza
      if(IsStopped())
         return;
      //--- ridisegna il chart ed attende 0.01 secondi
      ChartRedraw();
      Sleep(10);
     }
//--- 1 secondo di ritardo
   Sleep(1000);
//--- cambia il tipo di bordo
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- cambia il tipo di bordo
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- redisegna il chart ed attende per 1 secondo
   ChartRedraw();
   Sleep(1000);
//--- elimina l'etichetta
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- aspetta per 1 secondo
   Sleep(1000);
//---
  }
```

Proprietà degli oggetti

Gli oggetti grafici possono avere diverse proprietà in base al tipo di oggetto. I valori delle proprietà degli oggetti vengono impostati e ricevuti tramite la funzione corrispondente per lavorare con gli oggetti grafici.

Tutti gli oggetti utilizzati in analisi tecnica sono legati alle coordinate di tempo e prezzo: trendline, canali, gli strumenti di Fibonacci,etc. Ma c'è una serie di oggetti ausiliari destinati a migliorare l'interfaccia utente che sono legati alla parte sempre visibile di un grafico (finestre principali dei grafici o sottofinestre degli indicatori):

Oggetto	ID	X/Y	Larghezza/Altezza	Data/Prezzo	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	Si	—	Si	Si
Label	OBJ_LABEL	Si	Si (solo lettura)	—	Si	Si	Si
Button	OBJ_BUTTON	Si	Si	—	Si	—	—
Bitmap	OBJ_BITMAP	—	Si (solo lettura)	Si	—	Si	—
Bitmap Label	OBJ_BITMAP_LABEL	Si	Si (solo lettura)	—	Si	Si	—
Edit	OBJ_EDIT	Si	Si	—	Si	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	Si	Si	—	Si	—	—

Nella tabella vengono utilizzate le seguenti designazioni:

- **X/Y** — le coordinate specificate dei punti di ancoraggio relative ad un angolo del grafico;
- **Larghezza/Altezza** — gli oggetti hanno larghezza e altezza. Per "sola lettura", i valori di larghezza e altezza vengono calcolati solo dopo che l'oggetto è rappresentato sul grafico;
- **Data/Prezzo** — le coordinate del punto di ancoraggio sono specificate utilizzando i valori di data e prezzo;
- **OBJPROP_CORNER** — definisce l'angolo del grafico in base al quale sono specificate le coordinate del punto di ancoraggio. Può essere uno dei 4 valori dell'enumerazione ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** — definisce il punto di ancoraggio dell'oggetto stesso e può essere uno dei 9 valori dell'enumerazione ENUM_ANCHOR_POINT. Le coordinate in pixel sono specificate da questo punto all'angolo del grafico selezionato;
- **OBJPROP_ANGLE** — definisce l'angolo di rotazione dell'oggetto in senso antiorario.

Le funzioni che definiscono le proprietà degli oggetti grafici, nonché le operazioni ObjectCreate() e ObjectMove() per creare e spostare gli oggetti lungo il chart vengono effettivamente utilizzate per l'invio di comandi al chart. Se queste funzioni vengono eseguite correttamente, il comando viene

incluso nella coda comune degli eventi chart. Alterazioni visive nelle proprietà degli oggetti grafici vengono implementate quando si maneggia la coda degli eventi del grafico.

Quindi, non aspettatevi un aggiornamento visivo immediato di oggetti grafici dopo la chiamata di queste funzioni. In generale, gli oggetti grafici sul chart vengono aggiornati automaticamente dal terminale in seguito agli eventi di cambiamento - un nuovo arrivo di quotazione, il ridimensionamento della finestra chart, ecc. Usare la funzione ChartRedraw() per aggiornare con forza gli oggetti grafici.

Per le funzioni ObjectSetInteger() ed ObjectGetInteger()

ENUM_OBJECT_PROPERTY_INTEGER

Identificatore	Descrizione	Tipo di proprietà
OBJPROP_COLOR	Color	color
OBJPROP_STYLE	Style	ENUM_LINE_STYLE
OBJPROP_WIDTH	Spessore della linea	int
OBJPROP_BACK	Oggetto sullo sfondo	bool
OBJPROP_ZORDER	La priorità di un oggetto grafico per la ricezione degli eve	long

Identificatore	Descrizione	Tipo di proprietà
	nti cliccando su un grafico (CHARTEVENT_CLICK). Il valore zero predefinito viene impostato durante la creazione di un oggetto; la priorità può essere aumentata se necessario.	

Identificatore	Descrizione	Tipo di proprietà
	Quando si applica uno degli oggetti su un altro, solo uno di essi con la priorità più alta riceverà l'evento CHARTEVENT_CLICK.	
OBJPROP_FILL	Riempire un oggetto con il colore (per OBJ_RE	bool

Identificatore	Descrizione	Tipo di proprietà
	CTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	
OBJPROP_HIDDEN	Proibire mostrando il nome di un oggetto grafico all'interno della lista degli	bool

Identificatore	Descrizione	Tipo di proprietà
	oggetti dal menu del terminale "Grafici" - "Oggetti" - "Lista degli Oggetti". Il valore true consente di nascondere un oggetto dalla lista. Per impostazione predefinita, true	

Identificatore	Descrizione	Tipo di proprietà
	è impostato per gli oggetti che visualizzano gli eventi del calendario, la storia di trading ed agli oggetti creati da programmi MQL5. Per vedere tali oggetti grafici e accedere alle	

Identificatore	Descrizione	Tipo di proprietà
	loro proprietà, fare clic sul pulsante "Tutti" nella finestra "Lista degli oggetti".	
OBJPROP_SELECTED	L'oggetto è selezionato	bool
OBJPROP_READONLY	Possibilità di modificare il testo nell'oggetto Edit	bool
OBJPROP_TYPE	Tipo di oggetto	ENUM_OBJECT r/o

Identificatore	Descrizione	Tipo di proprietà
OBJPROP_TIME	Coordinate temporali	modificatore datetime=numero punto di ancoraggio
OBJPROP_SELECTABLE	Disponibilità oggetto	bool
OBJPROP_CREATETIME	Orario di creazione di oggetti	datetime r/o
OBJPROP_LEVELS	Numero di livelli	int
OBJPROP_LEVELCOLOR	Colore del livello-linea	color modifier=numero del livello
OBJPROP_LEVELSTYLE	Stile della linea-di-livello	ENUM_LINE_STYLE modifier=numero di livello
OBJPROP_LEVELWIDTH	Spessore della linea-di-	int modifier=numero di livello

Identificatore	Descrizione	Tipo di proprietà
	livello	
OBJPROP_ALIGN	Allineamento orizzontale del testo nell'oggetto "Edit" (OBJ_EDIT)	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	Dimensione carattere	int
OBJPROP_RAY_LEFT	Il raggio va a sinistra	bool
OBJPROP_RAY_RIGHT	Il raggio va a destra	bool
OBJPROP_RAY	Una linea verticale e pass	bool

Identificatore	Descrizione	Tipo di proprietà
	a attraverso tutte le finestre di un grafico	
OBJPROP_ELLIPSE	Mostra l'ellisse completa dell'oggetto Arco di Fibonacci (OBJ_FIBOARC)	bool
OBJPROP_ARROWCODE	Codice per l'oggetto Freccia	uchar
OBJPROP_TIMEFRAMES	La visibilità di un oggetto	impostazione dei flagsflags

Identificatore	Descrizione	Tipo di proprietà
	altimeframes	
OBJPROP_ANCHOR	Posizione del punto di ancoraggio di un oggetto grafico	ENUM_ARROW_ANCHOR (for OBJ_ARROW), ENUM_ANCHOR_POINT (for OBJ_LABEL, OBJ_BITMAP_LABEL and OBJ_TEXT)
OBJPROP_XDISTANCE	La distanza in pixel lungo l'asse X dal angolo vincolante (see note)	int
OBJPROP_YDISTANCE	La distanza in pixel lung	int

Identificatore	Descrizione	Tipo di proprietà
	o l'asse Y dall'angolo vincolante (see note)	
OBJPROP_DIRECTION	Trend dell'oggetto Gann	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	Livello di Elliott Wave Marcatura	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	Visualizzazione righe per creare la Elliott Wave	bool
OBJPROP_STATE	Status del	bool

Identificatore	Descrizione	Tipo di proprietà
	Bottone (premuto/non premuto)	
OBJPROP_CHART_ID	ID dell'oggetto "Chart" (OBJ_CHART). Esso permette di lavorare con le proprietà di questo oggetto, come con un chart normale e utilizzan	long r/o

Identificatore	Descrizione	Tipo di proprietà
	do le funzioni descritte in Operazioni Chart, ma ci sono delle eccezioni.	
OBJPROP_XSIZE	Larghezza dell'oggetto lungo l'asse X in pixel. Specificato per oggetti OBJ_LABEL (solo lettura)	int

Identificatore	Descrizione	Tipo di proprietà
	, OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_YSIZE	L'altezza dell'oggetto lungo l'asse Y in pixel. Specificato per oggetti OBJ	int

Identificatore	Descrizione	Tipo di proprietà
	_LABEL (solo lettura), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_XOFFSET	La coordinata X dell'angolo superiore sinistro dell	int

Identificatore	Descrizione	Tipo di proprietà
	a area visibile rettangolare negli oggetti grafici "Label Bitmap" e "Bitmap" (OBJ_BITMAP_LABEL e OBJ_BITMAP). Il valore è impostato in pixel rispetto all'angolo superio	

Identificatore	Descrizione	Tipo di proprietà
	resinistro dell'immagine originale.	
OBJPROP_YOFFSET	La coordinata Y dell'angolo superiore sinistro dell'area visibile rettangolare negli oggetti grafici "Label Bitmap" e "Bitmap" (OBJ_BITMAP	int

Identificatore	Descrizione	Tipo di proprietà
	_LABEL e OBJ_BITMAP). Il valore è impostato in pixel rispetto all'angolo superiore sinistro dell'immagine originale.	
OBJPROP_PERIOD	Timeframe per l'oggetto del Chart	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	Visualizzazione dell	bool

Identificatore	Descrizione	Tipo di proprietà
	a scala temporale per l'oggetto Chart	
OBJPROP_PRICE_SCALE	Visualizzazione della scala di prezzo per l'oggetto Chart	bool
OBJPROP_CHART_SCALE	La scala per l'oggetto Chart	int value in the range 0-5
OBJPROP_BGCOLOR	Il colore di sfondo per OBJ_EDIT, OBJ	color

Identificatore	Descrizione	Tipo di proprietà
	_BUTTON, OBJ_RECTANGLE_LABEL	
OBJPROP_CORNER	L'angolo del chart per collegare un oggetto grafico	ENUM_BASE_CORNER
OBJPROP_BORDER_TYPE	Tipo di bordo per l'oggetto "labelRectangle"	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	Colore del bordo per gli oggetti OBJ	color

Identificatore	Descrizione	Tipo di proprietà
	_EDIT e OBJ_BUTTON	

Quando si utilizzano operazioni del chart per l'oggetto "Chart" (OBJ_CHART), vengono inflitte le seguenti limitazioni:

- Non può essere chiuso con ChartClose();
- Simbolo/periodo non può essere modificato utilizzando la funzione ChartSetSymbolPeriod();
- Le seguenti proprietà sono inefficaci, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE e CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER).

È possibile impostare una modalità speciale di visualizzazione dell'immagine per gli oggetti OBJ_BITMAP_LABEL e OBJ_BITMAP. In questa modalità, viene visualizzata solo una parte dell'immagine originale (a cui è applicata una zona visibile rettangolare), mentre il resto dell'immagine diventa invisibile. La grandezza di questa area dovrebbe essere impostata utilizzando le proprietà OBJPROP_XSIZE e OBJPROP_YSIZE. L'area visibile può essere "spostata" solo all'interno l'immagine originale utilizzando le proprietà OBJPROP_XOFFSET e OBJPROP_YOFFSET.

For the fixed-sized objects: OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT and OBJ_CHART properties OBJPROP_XDISTANCE and OBJPROP_YDISTANCE set the position of the top left point of the object relative to the chart corner (OBJPROP_CORNER), from which the X and Y coordinates will be counted in pixels.

Per le funzioni ObjectSetDouble() e ObjectGetDouble()

ENUM_OBJECT_PROPERTY_DOUBLE

Identificatore	Descrizione	Tipo di proprietà
OBJPROP_PRICE	Coordinate Prezzo	double modifier=numero di punti di ancoraggio
OBJPROP_LEVELVALUE	Valore	double modifier=numero livello

Identificatore	Descrizione	Tipo di proprietà
	del livello	
OBJPROP_SCALE	Scala (proprietà degli oggetti di Gann e Fibonacci Arcs)	double
OBJPROP_ANGLE	Angolo. Per gli oggetti connessun angolo specificato, creato da un programma, il valore è pari a	double

Identificatore	Descrizione	Tipo di proprietà
	EMPTY_VALUE	
OBJPROP_DEVIATION	Deviazione per il Canale di Deviazione Standard	double

Per le funzioni ObjectSetString() e ObjectGetString()

ENUM_OBJECT_PROPERTY_STRING

Identificatore	Descrizione	Tipo di proprietà
OBJPROP_NAME	Nome oggetto	string
OBJPROP_TEXT	Descrizione dell'oggetto (il testo contenuto nell'ogg	string

Identificatore	Descrizione	Tipo di proprietà
	etto)	
OBJPROP_TOOLTIP	Il testo del tooltip. Se la proprietà non è impostata, allora viene visualizzato il tooltip generato automaticamente dal terminale. Un tooltip può essere disa	string

Identificatore	Descrizione	Tipo di proprietà
	bilitato impostando il valore "\n" (linefeed) adesso	
OBJPROP_LEVELTEXT	Descrizione del Livello	string modifier=numero livello
OBJPROP_FONT	Font	string
OBJPROP_BMPFILE	Il nome del file-BMP per Bitmap Label. Vedi anche Risorse	string modifier: 0-state ON, 1-state OFF
OBJPROP_SYMBOL	Simbolo per l'oggetto Chart	string

Per l' Oggetto OBJ_RECTANGLE_LABEL ("Label Rettangolo") può essere impostata una delle tre modalità di design, a cui i seguenti valori di ENUM_BORDER_TYPE corrispondono.

ENUM_BORDER_TYPE

Identificatore	Descrizione
BORDER_FLAT	Forma piatta
BORDER_RAISED	Forma sporgente
BORDER_SUNKEN	Forma concava

Per l'oggetto OBJ_EDIT ("Modifica") e per la funzione ChartScreenShot(), è possibile specificare il tipo di allineamento orizzontale utilizzando i valori dell'enumerazione ENUM_ALIGN_MODE.

ENUM_ALIGN_MODE

Identificatore	Descrizione
ALIGN_LEFT	L'allineamento a sinistra
ALIGN_CENTER	Centrato (solo per l'oggetto Edit)
ALIGN_RIGHT	Allineamento a destra

Esempio:

```
#define  UP          "\x0431"
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Oggetto",label_name," non trovato. Error code = ",GetLastError());
      //--- crea oggetto Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- imposta coordinate X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- imposta coordinate Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- definisce il colore del testo
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- definisce il testo per l'oggetto Label
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- definisce il font
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
```

```
      //--- definisce grandezza font
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
      //--- 45 gradi di rotazione in senso orario
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
      //--- disabilita il mouse per la selezione
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- lo disegna su un chart
      ChartRedraw(0);
     }
  }
```

Metodi di Binding Ogetti

Graphical objects Text, Label, Bitmap and Bitmap Label (OBJ_TEXT, OBJ_LABEL, OBJ_BITMAP and OBJ_BITMAP_LABEL) can have one of the 9 different ways of coordinate binding defined by the OBJPROP_ANCHOR property.

Object	ID	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	Yes	—	**Yes**	Yes
Label	OBJ_LABEL	Yes	Yes (read only)	—	Yes	**Yes**	Yes
Button	OBJ_BUTTON	Yes	Yes	—	Yes	—	—
Bitmap	OBJ_BITMAP	—	Yes (read only)	Yes	—	**Yes**	—
Bitmap Label	OBJ_BITMAP_LABEL	Yes	Yes (read only)	—	Yes	**Yes**	—
Edit	OBJ_EDIT	Yes	Yes	—	Yes	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	Yes	Yes	—	Yes	—	—

The following designations are used in the table:

- **X/Y** - coordinates of anchor points specified in pixels relative to a chart corner;
- **Width/Height** - objects have width and height. For "read only", the width and height values are calculated only once the object is rendered on chart;
- **Date/Price** - anchor point coordinates are specified using the date and price values;
- **OBJPROP_CORNER** - defines the chart corner relative to which the anchor point coordinates are specified. Can be one of the 4 values of the ENUM_BASE_CORNER enumeration;
- **OBJPROP_ANCHOR** - defines the anchor point in object itself and can be one of the 9 values of the ENUM_ANCHOR_POINT enumeration. Coordinates in pixels are specified from this very point to selected chart corner;
- **OBJPROP_ANGLE** - defines the object rotation angle counterclockwise.

La variante necessaria può essere specificata usando la funzione ObjectSetInteger(Chart_handle, object_name, OBJPROP_ANCHOR, Anchor_point_mode), dove anchor_point_mode è uno dei valori di ENUM_ANCHOR_POINT.

ENUM_ANCHOR_POINT

ID	Descrizione
ANCHOR_LEFT_UPPER	Punto di ancoraggio in alto a sinistra

ID	Descrizione
ANCHOR_LEFT	Punto di ancoraggio a sinistra nel centro
ANCHOR_LEFT_LOWER	Punto di ancoraggio nell'angolo in basso a sinistra
ANCHOR_LOWER	Punto di ancoraggio sotto sotto nel centro
ANCHOR_RIGHT_LOWER	Punto di ancoraggio nell'angolo in basso a destra
ANCHOR_RIGHT	Punto di ancoraggio a destra al centro
ANCHOR_RIGHT_UPPER	Punto di ancoraggio in alto a destra
ANCHOR_UPPER	Punto di ancoraggio sopra nel centro
ANCHOR_CENTER	Punto di ancoraggio strettamente nel centro dell'oggetto

The OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT and OBJ_CHART objects have a fixed anchor point in the upper left corner (ANCHOR_LEFT_UPPER).

Esempio:

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Object ",text_name," non trovato. Error code = ",GetLastError());
   //--- Ottiene il prezzo massimo del grafico
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- Crea etichetta dell'oggetto
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- Imposta colore del testo
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- Imposta colore background
   ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
   //--- Imposta il testo per l'oggetto grafico Etichetta
   ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
   //--- Imposta font del testo
   ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
   //--- Imposta la grandezza del font
   ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
   //--- Associazione all'angolo supriore destro
   ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
   //--- Ruota di 90 gradi in senso antiorario
   ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
   //--- Proibisce la selezione dell'oggetto dal mouse
   ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
   //--- ridisegna oggetto
   ChartRedraw(0);
  }
```

Gli oggetti grafici Freccia (OBJ_ARROW) hanno solo 2 modi di collegamento delle loro coordinate. Gli identificatori sono elencati in ENUM_ARROW_ANCHOR.

ENUM_ARROW_ANCHOR

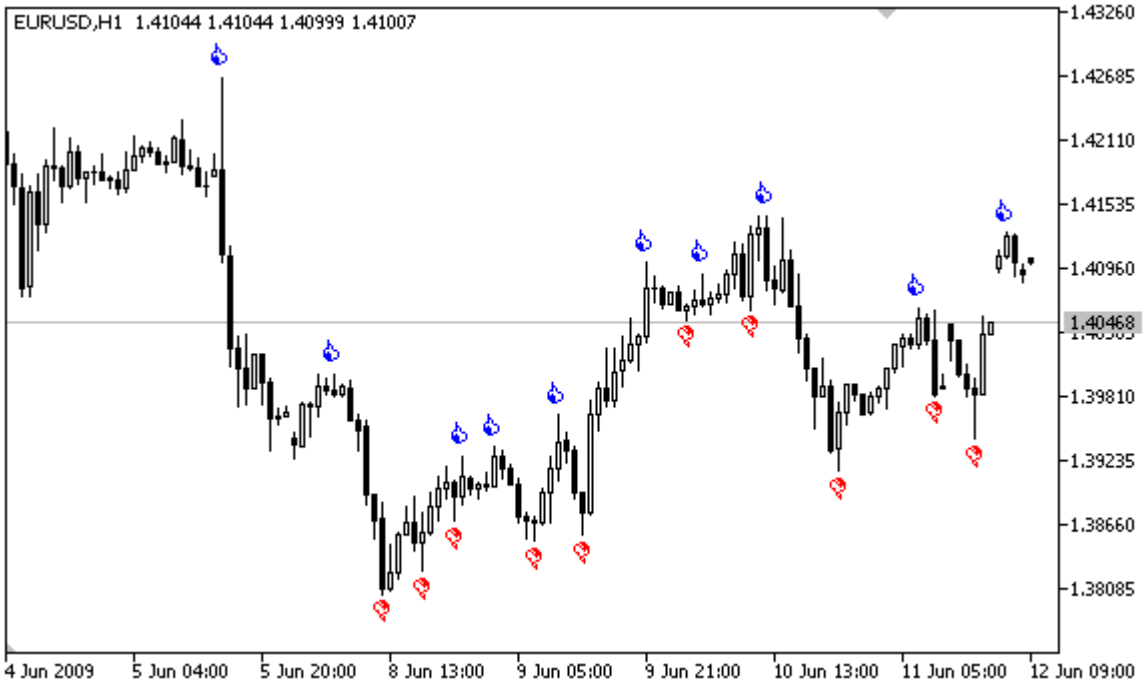
ID	Descrizione
ANCHOR_TOP	Ancoraggio sul lato superiore
ANCHOR_BOTTOM	Ancoraggio sul lato inferiore

Esempio:

```
voidOnStart()
  {
//--- Array ausiliari
   double Ups[],Downs[];
   datetime Time[];
//--- Imposta gli array come timeseries
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- Crea l'handle dell' Indicatore Fractals
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//-- Imposta il valore di Last error (ultimo errore) a Zero
   ResetLastError();
//--- Prova a copiare i valori dell'indicatore
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("Impossibile copiare i frattali superiori. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- Prova a copiare i valori dell'indicatore
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("Impossibile copiare i frattali inferiori. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- Copia le timeseries contenenti le aperture delle ultime 1000
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
      Print("Impossibile copiare i valori Opening Time delle ultime 1000 barre");
```

```
      return;
     }

  int upcounter=0,downcounter=0; // conta li il numero di frecce
  bool created;// riceve il risultato dei tentativi di creazione di un oggetto
  for(int i=2;i<copied;i++)// Esecuzione attraverso i valori dell'indicatore iFractal
    {
     if(Ups[i]!=EMPTY_VALUE)// Trova il frattale superiore
       {
        if(upcounter<10)// Crea non più di 10 frecce "Su"
          {
           //--- Prova a creare un oggetto "Su"
           created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
           if(created)// Se impostato - diamogli una regolata
             {
              //--- Il punto di ancoraggio è sotto per non coprire la barra
              ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
              //--- Tocco finale - disegnato
              ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
              upcounter++;
             }
          }
       }
     if(Downs[i]!=EMPTY_VALUE)// Trovato un frattale inferiore
       {
        if(downcounter<10)// Crea non più di 10 frecce "Giu"
          {
           //--- Prova a creare un oggetto "Giu"
           created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Down
           if(created)// Se impostato - diamogli una regolata
             {
              //--- Il punto di ancoraggio è sopra al fine di non coprire la barra
              ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
              //--- Tocco finale - disegnato
              ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
              downcounter++;
             }
          }
       }
    }
  }
```

Dopo l'esecuzione dello script il grafico sarà come in questa figura.



L'angolo del grafico a cui viene attaccato un oggetto

There is a number of graphical objects for which you can set a chart corner, relative to which the coordinates are specified in pixels. These are the following types of objects (in brackets object type identifiers are specified):

- Label (OBJ_LABEL);
- Button (OBJ_BUTTON);
- Bitmap Label (OBJ_BITMAP_LABEL);
- Edit (OBJ_EDIT).
- Rectangle Label (OBJ_RECTANGLE_LABEL);

Object	ID	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	Yes	–	Yes	Yes
Label	OBJ_LABEL	Yes	Yes (read only)	–	**Yes**	Yes	Yes
Button	OBJ_BUTTON	Yes	Yes	–	**Yes**	–	–
Bitmap	OBJ_BITMAP	–	Yes (read only)	Yes	–	Yes	–
Bitmap Label	OBJ_BITMAP_LABEL	Yes	Yes (read only)	–	**Yes**	Yes	–
Edit	OBJ_EDIT	Yes	Yes	–	**Yes**	–	–
Rectangle Label	OBJ_RECTANGLE_LABEL	Yes	Yes	–	**Yes**	–	–

The following designations are used in the table:

- **X/Y** - coordinates of anchor points specified in pixels relative to a chart corner;
- **Width/Height** - objects have width and height. For "read only", the width and height values are calculated only once the object is rendered on chart;
- **Date/Price** - anchor point coordinates are specified using the date and price values;
- **OBJPROP_CORNER** - defines the chart corner relative to which the anchor point coordinates are specified. Can be one of the 4 values of the ENUM_BASE_CORNER enumeration;
- **OBJPROP_ANCHOR** - defines the anchor point in object itself and can be one of the 9 values of the ENUM_ANCHOR_POINT enumeration. Coordinates in pixels are specified from this very point to selected chart corner;
- **OBJPROP_ANGLE** - defines the object rotation angle counterclockwise.

Per specificare l'angolo del grafico, da cui le coordinate X e Y vengono misurate in pixel, utilizzare ObjectSetInteger(ChartID, nome, OBJPROP_CORNER, **chart_corner**), Dove:

- chartID - identificatore del grafico;
- nome - nome dell' oggetto grafico;
- OBJPROP_CORNER - proprietà ID per specificare l'angolo per l'associazione;
- **chart_corner** - l'angolo del chart desiderato, può essere uno dei valori dell'enumerazione ENUM_BASE_CORNER.

ENUM_BASE_CORNER

ID	Descrizione
CORNER_LEFT_UPPER	Il centro delle coordinate è nell'angolo superiore sinistro del grafico
CORNER_LEFT_LOWER	Il centro delle coordinate è nell'angolo in basso a sinistra del grafico
CORNER_RIGHT_LOWER	Il centro di coordinate è nell'angolo in basso a destra del grafico
CORNER_RIGHT_UPPER	Il centro di coordinate è nell'angolo in alto a destra del grafico

Esempio:

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 int    anchor_point,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   if(ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0))
     {
      ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner);
      ObjectSetInteger(chart_id,name,OBJPROP_ANCHOR,anchor_point);
      ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
      ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
      ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
     }
   else
      Print("Failed to create the object OBJ_LABEL ",name,", Error code = ", GetLastEr
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int height=(int)ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
```

```
   int width=(int)ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
   CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,ANCHOR_LEFT_UPPER,arrows[0],50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,ANCHOR_RIGHT_UPPER,arrows[1],50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,ANCHOR_RIGHT_LOWER,arrows[2],50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,ANCHOR_LEFT_LOWER,arrows[3],50,50);
  }
```

Visibilità degli oggetti

La combinazione dei flag di visibilità degli oggetti determina i timeframes del grafico, in cui l'oggetto è visibile. Per impostare/ottenere il valore della proprietà OBJPROP_TIMEFRAMES, è possibile utilizzare le funzioni ObjectSetInteger()/ObjectGetInteger().

ID	Valore	Descrizione
OBJ_NO_PERIODS	0	L'oggetto non viene disegnato in tutti i timeframes
OBJ_PERIOD_M1	0x00000001	L'oggetto viene disegnato nel grafico 1-minuto
OBJ_PERIOD_M2	0x00000002	L'oggetto viene disegnato nel grafico 2-minuti
OBJ_PERIOD_M3	0x00000004	L'oggetto viene disegnato nel grafico 3-minuti
OBJ_PERIOD_M4	0x00000008	L'oggetto viene disegnato nel grafico 4-minuti
OBJ_PERIOD_M5	0x00000010	L'oggetto viene disegnato nel grafico 5-minuti
OBJ_PERIOD_M6	0x00000020	L'oggetto viene disegnato nel grafico 6-minuti
OBJ_PERIOD_M10	0x00000040	L'oggetto viene disegnato nel grafico 10-minuti
OBJ_PERIOD_M12	0x00000080	L'oggetto viene disegnato nel grafico 12-minuti
OBJ_PERIOD_M15	0x00000100	L'oggetto viene disegnato nel grafico 2-minuti
OBJ_PERIOD_M20	0x00000200	L'oggetto viene disegnato nel grafico 20-minuti
OBJ_PERIOD_M30	0x00000400	L'oggetto viene disegnato nel grafico 30-minuti
OBJ_PERIOD_H1	0x00000800	L'oggetto viene disegnato nel grafico 1-ora
OBJ_PERIOD_H2	0x00001000	L'oggetto viene disegnato nel grafico 2-ore
OBJ_PERIOD_H3	0x00002000	L'oggetto viene disegnato nel grafico 3-ore
OBJ_PERIOD_H4	0x00004000	L'oggetto viene disegnato nel grafico 4-ore

ID	Valore	Descrizione
OBJ_PERIOD_H6	0x00008000	L'oggetto viene disegnato nel grafico 6-ore
OBJ_PERIOD_H8	0x00010000	L'oggetto viene disegnato nel grafico 8-ore
OBJ_PERIOD_H12	0x00020000	L'oggetto viene disegnato nel grafico 12-ore
OBJ_PERIOD_D1	0x00040000	L'oggetto viene disegnato nel grafico giornaliero
OBJ_PERIOD_W1	0x00080000	L'oggetto viene disegnato nel grafico settimanale
OBJ_PERIOD_MN1	0x00100000	L'oggetto viene disegnato nel grafico mensile
OBJ_ALL_PERIODS	0x001fffff	L'oggetto viene disegnato in tutti i timeframes

La visibilità delle flags può essere combinata con il simbolo "|", per esempio, la combinazione di flag OBJ_PERIOD_M10 | OBJ_PERIOD_H4 significa che l'oggetto sarà visibile sui timeframes a 10-minuti e 4-ore.

Esempio:

```
voidOnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // Il Massimo(High) del giorno precedente
   double prevLow;            // Il Minimo(Low) del giorno precedente
   double highs[],lows[];     // Array per Massimo e Minimo

//--- Resetta l'ultimo errore
   ResetLastError();
//--- Riceve gli ultimi 2 valori Massimi sul timeframe giornaliero
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // Se la copia ha avuto successo
     {
      Print("I prezzi Massimi per gli ultimi 2 giorni sono stati ottenuti con successo
      prevHigh=highs[0]; // Il Massimo(High) del giorno precedente
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // Oggetto con il nome highlevel non trovato
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // Crea l'oggetto Linea Orizzontal
        }
      //--- Imposta il valore per il livello del prezzo per la linea highlevel
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
```

```
      //--- Imposta la visibilità solo per PERIOD_M10 e PERIOD_H4
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("Non posso ottenere i prezzi Massimi(High) per gli ultimi 2 giorni, Errore
     }

//--- Resetta l'ultimo errore
   ResetLastError();
//--- Ottiene i valori Minimi(Low) di 2 giorni sul timeframe giornaliero
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // Se la copia ha avuto successo
     {
      Print("I prezzi Minimi(Low) per gli ultimi 2 giorni sono stati ottenuti con succ
      prevLow=lows[0]; // Il Minimo del giorno precedente
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // Oggetto con il nome lowlevel non trovato
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // Crea l'oggetto Linea Orizzontale
        }
      //--- Imposta il valore per il livello del prezzo per la linea lowlevel
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- Imposta la visibilità solo per PERIOD_M10 e PERIOD_H4
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("Non posso ottenere i prezzi Minimi(Low) per gli ultimi 2 giorni, Errore

   ChartRedraw(0); // aggiorna il grafico forzatamente
  }
```

Vedi anche

PeriodSeconds, Period, Timeframes dei Grafici, Data ed Ora

Livelli delle Elliott Wave (_* Onde di Elliot)

Le onde di Elliott sono rappresentate da due oggetti grafici di tipo OBJ_ELLIOTWAVE5 e OBJ_ELLIOTWAVE3. Per impostare le dimensioni dell' onda (metodo di etichettatura onda), viene utilizzata la proprietà OBJPROP_DEGREE, per cui uno dei valori dell'enumerazione ENUM_ELLIOT_WAVE_DEGREE può essere assegnato.

ENUM_ELLIOT_WAVE_DEGREE

ID	Descrizione
ELLIOTT_GRAND_SUPERCYCLE	Grand Supercycle
ELLIOTT_SUPERCYCLE	Supercycle
ELLIOTT_CYCLE	Ciclo
ELLIOTT_PRIMARY	Primario
ELLIOTT_INTERMEDIATE	Intermedio
ELLIOTT_MINOR	Minore
ELLIOTT_MINUTE	Minuto
ELLIOTT_MINUETTE	Minuette
ELLIOTT_SUBMINUETTE	Subminuette

Esempio:

```
   for(int i=0;i<ObjectsTotal(0);i++)
     {
      string currobj=ObjectName(0,i);
      if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
         ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
        {
         //--- imposta i livelli di marking in INTERMEDIATE
         ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
         //--- mostra linee tra i tops delle onde
         ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
         //--- imposta il colore della linea
         ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
         //--- imposta lo spessore della linea
         ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
         //--- imposta descrizione
         ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
        }
     }
```

Oggetti Gann

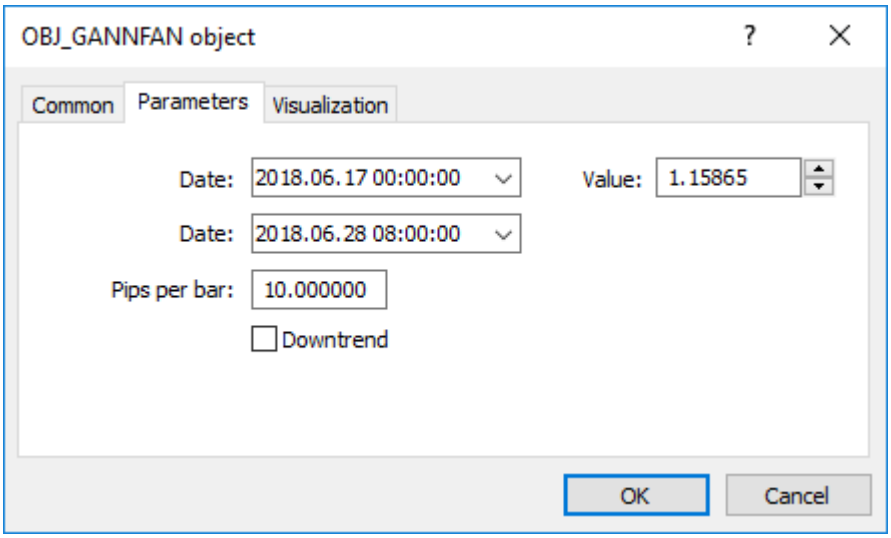
Per gli oggetti Gann Fan (OBJ_GANNFAN) e le Gann Grid (OBJ_GANNGRID) è possibile specificare due valori dell'enumerazione ENUM_GANN_DIRECTION che impostano la direzione del trend.

ENUM_GANN_DIRECTION

ID	Descrizione
GANN_UP_TREND	Linea corrispondente alla linea di trend rialzista
GANN_DOWN_TREND	Linea corrispondente alla tendenza al ribasso

Per impostare la scala della linea principale, come 1x1, utilizzare la funzione ObjectSetDouble(Chart_handle, gann_object_name, OBJPROP_SCALE, Scala), dove:

- chart_handle - finestra del grafico in cui si trova l'oggetto;
- gann_object_name - nome oggetto;
- OBJPROP_SCALE - identificatore della proprietà "Scala";
- scale - scala richiesta in unità di Pips/Bar.



Esempio di creazione di Gann Fan:

```
voidOnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// Oggetto non trovato
     {
      //--- Informa riguardo il fallimento
      Print("Object ",my_gann," non trovato. Error code = ",GetLastError());
      //--- Ottiene il prezzo massimo del grafico
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- Ottiene il prezzo minimo del grafico
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
      //--- Quante barre vengono mostrate nel grafico?
```

```
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- Crea un array, per scrivere il tempo di apertura di ogni barra
      datetime Time[];
      //--- Dispone l' accesso al array come quello delle TimeSeries
      ArraySetAsSeries(Time,true);
      //--- Ora copia i dati delle barre visibili nel grafico in questo array
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("Non posso copiare l'array con l'orario di apertura!");
         return;
        }
      //--- Preparazioni preliminari completate

      //--- Indice deppa barra centrale nel grafico
      int center_bar=bars_on_chart/2;
      //--- Equatore del grafico - tra il massimo ed il minimo
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- Imposta le coordinate del primo ancor point al centro
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- Secondo anchor point a destra
                   Time[center_bar/2],(mean+chart_min_price)/2.0);
      Print("Time[center_bar] = "+(string)Time[center_bar]+"  Time[center_bar/2] = "+
      //Print("Time[center_bar]/="+Time[center_bar]+"  Time[center_bar/2]="+Time[cente
      //--- Imposta la scala in unità per Pips / Bar
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- Imposta il line trend
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- Imposta lo spessore della linea
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- Definisce lo stile della linea
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- Imposta il colore della linea
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- Permette all'utente di selezionare un oggetto
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- Selezionalo da solo
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- Disegnalo sul grafico
      ChartRedraw(0);
     }
  }
```

I Web Colors

Le costanti seguenti colori sono definite per il tipo color:

clrBlack	clrDarkGreen	clrDarkSlateGray	clrOlive	clrGreen	clrTeal	clrNavy	clrPurple
clrMaroon	clrIndigo	clrMidnightBlue	clrDarkBlue	clrDarkOliveGreen	clrSaddleBrown	clrForestGreen	clrOliveDrab
clrSeaGreen	clrDarkGoldenrod	clrDarkSlateBlue	clrSienna	clrMediumBlue	clrBrown	clrDarkTurquoise	clrDimGray
clrLightSeaGreen	clrDarkViolet	clrFireBrick	clrMediumVioletRed	clrMediumSeaGreen	clrChocolate	clrCrimson	clrSteelBlue
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen	clrCadetBlue	clrDarkOrchid	clrYellowGreen	clrLimeGreen	clrOrangeRed
clrDarkOrange	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue	clrBlue	clrMagenta	clrRed	clrGray	clrSlateGray	clrPeru	clrBlueViolet
clrLightSlateGray	clrDeepPink	clrMediumTurquoise	clrDodgerBlue	clrTurquoise	clrRoyalBlue	clrSlateBlue	clrDarkKhaki
clrIndianRed	clrMediumOrchid	clrGreenYellow	clrMediumAquamarine	clrDarkSeaGreen	clrTomato	clrRosyBrown	clrOrchid
clrMediumPurple	clrPaleVioletRed	clrCoral	clrCornflowerBlue	clrDarkGray	clrSandyBrown	clrMediumSlateBlue	clrTan
clrDarkSalmon	clrBurlyWood	clrHotPink	clrSalmon	clrViolet	clrLightCoral	clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

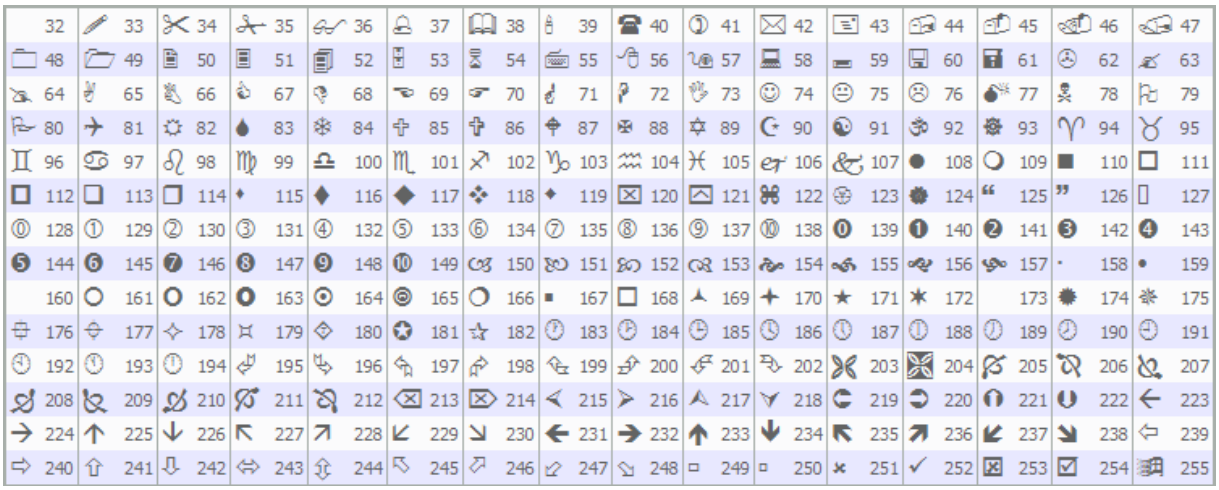
Il colore può essere impostato su un oggetto utilizzando la funzione ObjectSetInteger(). Per impostare il colore di indicatori personalizzati viene utilizzata la funzione PlotIndexSetInteger(). Per ottenere valori del colore ci sono funzioni simili ObjectGetInteger() e PlotIndexGetInteger().

Esempio:

```
//---- impostazioni indicatore
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_type3   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_color2  clrRed
#property indicator_color3  clrLime
```

Wingdings

I caratteri di Wingdings utilizzati con l'oggetto OBJ_ARROW:



Un carattere necessario può essere impostato con la funzione ObjectSetInteger().

Esempio:

```
voidOnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);      // Ottiene il prezzo di C
//--- Se il prezzo è stato ottenuto
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // Crea una freccia
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // Imposta il codice dell
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // Imposta orario
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// Imposta prezzo
      ChartRedraw(0);                                         // Disegna ora la freccia
     }
   else
      Print("Impossibile ottenere l'ultimo prezzo di Chiusura(Close)!");
  }
```

Indicators Constants

Ci sono 37 indicatori tecnici predefiniti, che possono essere usati in programmi scritti nel linguaggio MQL5. Inoltre, vi è la possibilità di creare indicatori personalizzati utilizzando la funzione iCustom(). Tutte le costanti richieste per tale sono divise in 5 gruppi:

- Costanti di prezzo - per selezionare il tipo di prezzo o volume, in cui viene calcolato un indicatore;
- Metodi di smoothing - metodi di smoothing built-in utilizzati negli indicatori;
- Linee indicatore - Identificatori di buffer indicatore quando si accede utilizzando i valori degli indicatori CopyBuffer();
- Gli stili di disegno - per indicare uno dei 18 tipi di disegno ed impostare lo stile di disegno linea;
- Proprietà indicatori personalizzati sono utilizzate in funzioni per lavorare con indicatori custom;
- I tipi di indicatori sono utilizzati per specificare il tipo di indicatore tecnico durante la creazione di un handle con IndicatorCreate();
- Identificatori di tipi di dati vengono utilizzati per specificare il tipo di dati passati in un array di tipo MqlParam nella funzione IndicatorCreate().

Price Constants

I calcoli di indicatori tecnici richiedono valori di prezzo e/o valori di volumi, sui quali i calcoli verranno eseguiti. Ci sono 7 identificatori predefiniti dall'enumerazione ENUM_APPLIED_PRICE, utilizzati per specificare il prezzo base per il calcolo desiderato.

ENUM_APPLIED_PRICE

ID	Descrizione
PRICE_CLOSE	Prezzo di chiusura
PRICE_OPEN	Prezzo di apertura
PRICE_HIGH	Il prezzo massimo per il periodo
PRICE_LOW	Il prezzo minimo per il periodo
PRICE_MEDIAN	Prezzo mediano, (high + low)/2
PRICE_TYPICAL	Prezzo tipico, (high + low + close)/3
PRICE_WEIGHTED	Prezzo medio, (alto + basso + vicino + close)/4

Se il volume viene utilizzata nei calcoli, è necessario specificare uno dei due valori dell'enumerazione ENUM_APPLIED_VOLUME.

ENUM_APPLIED_VOLUME

ID	Descrizione
VOLUME_TICK	Tick volume
VOLUME_REAL	Trade volume

L'indicatore tecnico iStochastic() può essere calcolato in due modi diversi per:

- o solo i prezzi Close;
- o prezzi High e Low.

Per selezionare una variante necessaria per il calcolo, specificare uno dei valori dell'enumerazione ENUM_STO_PRICE.

ENUM_STO_PRICE

ID	Descrizione
STO_LOWHIGH	Il calcolo si basa sui prezzi inferiore/superiore
STO_CLOSECLOSE	Calculation is based on Close/Close prices

Se un indicatore tecnico usa per i calcoli i dati prezzo, il tipo di cui viene impostato ENUM_APPLIED_PRICE, quindi la gestione di qualsiasi indicatore (costruita nel terminale o scritta da un utente) può essere utilizzata come la serie prezzo di ingresso. In questo caso, i valori del buffer zero del indicatore verranno utilizzati per i calcoli. In questo modo è facile costruire i valori di un indicatore utilizzando i valori di un altro indicatore. L'handle di un indicatore personalizzato viene creato chiamando la funzione iCustom().

Esempio:

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- parametri di input
input int      RSIperiod=14;         // Periodo per il calcolo dell'RSI
input int      Smooth=8;             // Periodo smussamento RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // Metodo di smussamento
//---- disegna RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- disegna RSI_Smoothed
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- buffers indicatore
double         RSIBuffer[];          // Qui conserviamo i valori di RSI
double         RSI_SmoothedBuffer[]; // Qui ci saranno i valori smussati di RSI
int            RSIhandle;            // Handle dell' indicatore RSI
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
   IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
   RSIhandle=iRSI(NULL,0,RSIperiod,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
                )

  {
```

```
//---  Resetta il valore dell'ultimo errore
   ResetLastError();
//--- Riceve i dati dell'indicatore RSI in un array RSIBuffer []
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("Impossibile copiare i valori dell'indicatore RSI. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- Crea l'indicatore dei valori medi con i valori di RSI
   int RSI_MA_handle=iMA(NULL,0,Smooth,0,meth,RSIhandle);
   copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
   if(copied<=0)
     {
      Print("Impossibile copiare l'indicatore smussato di RSI. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Metodi di smussamento

Molti indicatori tecnici si basano su vari metodi di smussamento della serie di prezzi. Alcuni indicatori tecnici standard richiedono la specificazione del tipo smussamento come parametro di input. Per specificare il tipo desiderato di smussamento, vengono utilizzati identificatori elencati nell'enumerazione ENUM_MA_METHOD.

ENUM_MA_METHOD

ID	Descrizione
MODE_SMA	Media semplice
MODE_EMA	Media esponenziale
MODE_SMMA	Media smussata
MODE_LWMA	Media lineare-ponderata

Esempio:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//--- handles per medie mobili
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- ottenere gli handle di MA
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

Linee Indicatori

Alcuni indicatori tecnici hanno diversi buffers disegnati nel grafico. La numerazione dei buffer indicatori inizia con 0. Durante la copia di valori degli indicatori che utilizzano la funzione CopyBuffer() in un array del tipo double, per alcuni indicatori si può indicare l'identificatore di un buffer copiato anziché il suo numero.

Identificatori di linee indicatore ammissibili quando si copiano i valori di iMACD(),iRV() e iStochastic().

Constant	Valore	Descrizione
MAIN_LINE	0	Linea principale
SIGNAL_LINE	1	Linea di segnale

Identificatori di linee indicatore ammissibili quando si copiano i valori di ADX() e ADXW().

Constant	Valore	Descrizione
MAIN_LINE	0	Linea principale
PLUSDI_LINE	1	Linea + DI
MINUSDI_LINE	2	Linea -DI

Identificatori di linee indicatore ammissibili quando si copiano i valori di iBands().

Constant	Valore	Descrizione
BASE_LINE	0	Linea principale
UPPER_BAND	1	Limite superiore
LOWER_BAND	2	Limite inferiore

Identificatori di linee indicatore ammissibili quando si copiano i valori di iEnvelopes() e iFractals().

Constant	Valore	Descrizione
UPPER_LINE	0	Linea superiore
LOWER_LINE	1	Linea inferiore

Identificatori di linee indicatore ammissibili quando si copiano i valori di iGator()

Constant	Valore	Descrizione
UPPER_HISTOGRAM	0	Istogramma superiore
LOWER_HISTOGRAM	2	Istogramma inferiore

Identificatori di linee indicatore ammissibili quando si copiano i valori di iAlligator().

Constant	Valore	Descrizione
GATORJAW_LINE	0	Linea Jaw
GATORTEETH_LINE	1	Linea Teeth
GATORLIPS_LINE	2	Linea Lips

Identificatori di linee indicatore ammissibili quando si copiano i valori di iIchimoku().

Constant	Valore	Descrizione
TENKANSEN_LINE	0	Linea Tenkan-sen
KIJUNSEN_LINE	1	Linea Kijun-sen
SENKOUSPANA_LINE	2	Linea Senkou Span A
SENKOUSPANB_LINE	3	Linea Senkou Span B
CHIKOUSPAN_LINE	4	Linea Chikou Span

Drawing Styles

Quando si crea un indicatore personalizzato, è possibile specificare uno dei 18 tipi di grafico tracciato (come visualizzato nella finestra del grafico principale o una sottofinestra grafico), i cui valori sono specificati nella enumerazione ENUM_DRAW_TYPE.

In un indicatore personalizzato, è consentito utilizzare qualsiasi tipo di costruzione/disegno di indicatore. Ogni tipo di costruzione richiede la specifica di uno a cinque array globali per memorizzare i dati necessari per il disegno. Questi array di dati devono essere legati con il buffer indicatore utilizzando la funzione SetIndexBuffer(). Il tipo di dati da ENUM_INDEXBUFFER_TYPE deve essere specificato per ogni buffer.

A seconda dello stile di disegno, potrebbe essere necessario 1-4 valori buffer (contrassegnato come INDICATOR_DATA). Se uno stile ammette l'alternarsi dinamico dei colori (tutti gli stili contengono COLORE nei loro nomi), quindi avrete bisogno di un altro buffer di colore (tipo indicato INDICATOR_COLOR_INDEX). I buffer di colore sono sempre tenuti dopo buffer valore corrispondente allo stile.

ENUM_DRAW_TYPE

ID	Descrizione	Data buffers	Color buffers
DRAW_NONE	Non disegnato	1	0
DRAW_LINE	Linea	1	0
DRAW_SECTION	Sezione	1	0
DRAW_HISTOGRAM	Istogramma dalla linea dello zero	1	0
DRAW_HISTOGRAM2	Istogramma dei due buffer indicatore	2	0
DRAW_ARROW	Disegnamento frecce	1	0
DRAW_ZIGZAG	Stili Zigzag permettono la sezione verticale	2	0

ID	Descrizione	Data buffers	Color buffers
	sulla barra		
DRAW_FILLING	Color fill between the two levels	2	0
DRAW_BARS	Visualizzare come una sequenza di barre	4	0
DRAW_CANDLES	Visualizza come una sequenza di candele	4	0
DRAW_COLOR_LINE	Linea multicolore	1	1
DRAW_COLOR_SECTION	Sezione multicolore	1	1
DRAW_COLOR_HISTOGRAM	Istogramma multicolore dalla linea dello zero	1	1
DRAW_COLOR_HISTOGRAM2	Istogramma multicolore dei due buffer indicatore	2	1
DRAW_COLOR_ARROW	Disegno frecce multicolori	1	1

ID	Descrizione	Data buffers	Color buffers
DRAW_COLOR_ZIGZAG	ZigZag multicolore	2	1
DRAW_COLOR_BARS	Multicolore barre	4	1
DRAW_COLOR_CANDLES	Candele multicolore	4	1

Per perfezionare la visualizzazione del tipo di disegno selezionato, vengono utilizzati identificatori elencati in ENUM_PLOT_PROPERTY.

Per le funzioni PlotIndexSetInteger() e PlotIndexGetInteger()

ENUM_PLOT_PROPERTY_INTEGER

ID	Descrizione	Tipo di proprietà
PLOT_ARROW	Codice Arrow per DRAW_ARROW stile	uchar
PLOT_ARROW_SHIFT	Spostamento verticale di frecce per stile DRAW_ARROW	int
PLOT_DRAW_BEGIN	Numero di barre iniziali, senza disegno e valori nel DataWindow	int
PLOT_DRAW_TYPE	Tipo di costruzione grafica	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	Segno di visualizzazione dei valori di costruzione nel DataWindow	bool
PLOT_SHIFT	Spostamento dell'indicatore tracciato lungo l'asse del tempo nelle bars	int

ID	Descrizione	Tipo di proprietà
PLOT_LINE_STYLE	Stile di Disegno della linea	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	Lo spessore della linea di disegno	int
PLOT_COLOR_INDEXES	Il numero di colori	int
PLOT_LINE_COLOR	L'indice di un tampone contenente il colore di disegno	color modifier = index number of colors

Per la funzione PlotIndexSetDouble()

ENUM_PLOT_PROPERTY_DOUBLE

ID	Descrizione	Tipo di proprietà
PLOT_EMPTY_VALUE	Un valore vuoto per la stampa, per cui non esiste nessun disegno	double

Per la funzione PlotIndexSetString()

ENUM_PLOT_PROPERTY_STRING

ID	Descrizione	Tipo di proprietà
PLOT_LABEL	Il nome della serie grafica dell' indicatore da visualizzare nel DataWindow. Quando si lavora con stili grafici complessi che richiedono diversi buffer indicatori per la visualizzazione, i nomi per ogni buffer possono essere specificati utilizzando ";" come separatore. Codice di esempio è mostrato in DRAW_CANDLES	string

5 stili possono essere utilizzati per disegnare linee in indicatori personalizzati. Essi sono validi solo per lo spessore della linea 0 o 1.

ENUM_LINE_STYLE

ID	Descrizione
STYLE_SOLID	Linea continua
STYLE_DASH	Linea spezzata
STYLE_DOT	Linea tratteggiata
STYLE_DASHDOT	Linea punteggiata-tratteggiata
STYLE_DASHDOTDOT	Tratteggio - due punti

Per impostare lo stile di disegno linea ed il tipo di disegno, viene utilizzata la funzione PlotIndexSetInteger(). Per le Estensioni di Fibonacci lo spessore e lo stile di disegno dei livelli può essere indicato con la funzione ObjectSetInteger().

Esempio:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- buffers indicatore
double         MABuffer[];
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Associa l'Array al buffer indicatore con indice 0
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- Imposta la linea di disegno
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- Imposta la linea di stile
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- Imposta colore della linea
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- Imposta lo spessore della linea
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- Imposta etichette per la linea
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

```
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
     {
      MABuffer[i]=close[i];
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Proprietà Indicatori Personalizzati

Il numero di buffer indicatore che può essere utilizzato in un indicatore personalizzato è illimitato. Ma per ogni array, che è designato come il buffer indicatore utilizzando la funzione SetIndexBuffer(), è necessario specificare il tipo di dati che verranno memorizzati. Questo può essere uno dei valori dell'enumerazione ENUM_INDEXBUFFER_TYPE.

ENUM_INDEXBUFFER_TYPE

ID	Descrizione
INDICATOR_DATA	Dati per disegnare
INDICATOR_COLOR_INDEX	Color
INDICATOR_CALCULATIONS	Buffer ausiliari per calcoli intermedi

Un indicatore personalizzato ha un sacco di impostazioni per fornire visualizzazioni convenienti. Queste impostazioni vengono effettuate attraverso l'assegnazione d proprietà indicatore corrispondente utilizzando le funzioni IndicatorSetDouble(), IndicatorSetInteger() e IndicatorSetString(). Identificatori di proprietà indicatori sono elencati nell'enumerazione ENUM_CUSTOMIND_PROPERTY.

ENUM_CUSTOMIND_PROPERTY_INTEGER

ID	Descrizione	Tipo di proprietà
INDICATOR_DIGITS	Precisione del disegno di valori degli indicatori	int
INDICATOR_HEIGHT	Altezza fissa della finestra dell'i	int

ID	Descrizione	Tipo di proprietà
	ndicatore (il comando del preprocessore #property indicator_height)	
INDICATOR_LEVELS	Numero di livelli nella finestra dell'indicatore	int
INDICATOR_LEVELCOLOR	Colore della linea di livello	color modifier = level number
INDICATOR_LEVELSTYLE	Stile della linea di	ENUM_LINE_STYLE modifier = level number

ID	Descrizione	Tipo di proprietà
	livello	
INDICATOR_LEVELWIDTH	Spessore della linea di livello	int modifier = level number
INDICATOR_FIXED_MINIMUM	Minimo Fisso per la finestra dell'indicatore. La proprietà può essere scritta solo dalla funzione IndicatorSetInteger()	bool
INDICATOR_FIXED_MAXIMUM	Massim	bool

ID	Descrizione	Tipo di proprietà
	o Fisso per la finestra dell'indicatore. La proprietà può essere scritta solo dalla funzione IndicatorSetInteger()	

ENUM_CUSTOMIND_PROPERTY_DOUBLE

ID	Descrizione	Tipo di proprietà
INDICATOR_MINIMUM	Minima della finestra dell'indicatore	double

ID	Descrizione	Tipo di proprietà
INDICATOR_MAXIMUM	Massima della finestra dell'indicatore	double
INDICATOR_LEVELVALUE	Valore del livello	double modifier = level number

ENUM_CUSTOMIND_PROPERTY_STRING

ID	Descrizione	Tipo di proprietà
INDICATOR_SHORTNAME	Nome breve dell'indicatore	string
INDICATOR_LEVELTEXT	Descrizione del Livello	string modifier = level number

Esempi:

```
//--- impostazioni indicatore
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_color2  clrRed
//--- parametri di input
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- buffers indicatore
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- imposta precisione
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- imposta livelli
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- imposta massimo e minimo per sottofinestra
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- imposta la prima barra da quell' indice sarà disegnato
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- imposta stile STYLE_DOT per seconda linea
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- nome per DataWindow e indicatore di etichetta sottofinestra
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- imposta il valore di disegno della linea come vuoto
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- inizializzazione fatta
  }
```

Tipi di indicatori tecnici

Ci sono due modi per creare una handle indicatore per ulteriori accesso ad i suoi valori. Il primo modo è quello di specificare direttamente un nome di funzione dall'elenco degli indicatori tecnici. Il secondo metodo che utilizza l' IndicatorCreate() è quello di creare in modo uniforme un handle di qualsiasi indicatore tramite l'assegnazione di un identificatore dell'enumerazione ENUM_INDICATOR. Entrambi i modi di creazione handle sono uguali, è possibile utilizzare quello che è più conveniente in un caso particolare, quando si scrive un programma in MQL5.

Quando si crea un indicatore di tipo IND_CUSTOM, il campo *tipo* del primo elemento di un array di parametri di input MqlParam deve avere il valore TYPE_STRING dell'enumerazione ENUM_DATATYPE, mentre il campo *valore_stringa* del primo elemento deve contenere il nome dell'indicatore personalizzato.

ENUM_INDICATOR

Identificatore	Indicatore
IND_AC	Accelerator Oscillator
IND_AD	Accumulation/Distribution
IND_ADX	Average Directional Index
IND_ADXW	ADX by Welles Wilder
IND_ALLIGATOR	Alligator
IND_AMA	Adaptive Moving Average
IND_AO	Awesome Oscillator
IND_ATR	Average True Range
IND_BANDS	Bollinger Bands®
IND_BEARS	Bears Power
IND_BULLS	Bulls Power
IND_BWMFI	Market Facilitation Index
IND_CCI	Commodity Channel Index
IND_CHAIKIN	Chaikin Oscillator
IND_CUSTOM	Custom indicator
IND_DEMA	Double Exponential Moving Average
IND_DEMARKER	DeMarker
IND_ENVELOPES	Envelopes
IND_FORCE	Force Index
IND_FRACTALS	Fractals
IND_FRAMA	Fractal Adaptive Moving Average

Identificatore	Indicatore
IND_GATOR	Gator Oscillator
IND_ICHIMOKU	Ichimoku Kinko Hyo
IND_MA	Moving Average
IND_MACD	MACD
IND_MFI	Money Flow Index
IND_MOMENTUM	Momentum
IND_OBV	On Balance Volume
IND_OSMA	OsMA
IND_RSI	Relative Strength Index
IND_RVI	Relative Vigor Index
IND_SAR	Parabolic SAR
IND_STDDEV	Standard Deviation
IND_STOCHASTIC	Stochastic Oscillator
IND_TEMA	Triple Exponential Moving Average
IND_TRIX	Triple Exponential Moving Averages Oscillator
IND_VIDYA	Variable Index Dynamic Average
IND_VOLUMES	Volumes
IND_WPR	Williams' Percent Range

Identificatori Tipo di Dati

Quando si crea un handle indicatore utilizzando la funzione IndicatorCreate(), un array di tipo MqlParam deve essere specificato come ultimo parametro. Di conseguenza, la struttura MqlParam, descrivendo l'indicatore, contiene un campo speciale *type*. Questo campo contiene le informazioni sul tipo di dati (real, integer o string) che vengono passati da un particolare elemento della matrice. Il valore di questo campo della struttura MqlParam può essere uno dei valori ENUM_DATATYPE.

ENUM_DATATYPE

Identificatore	Tipo di dati
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	color
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

Ciascun elemento della matrice descrive il corrispondente parametro input di un indicatore tecnico creato, quindi il tipo e l'ordine degli elementi dell'array deve essere rigorosamente mantenuto in conformità con la descrizione.

Environment State

Le costanti che descrivono l'attuale ambiente di runtime di un programma-mql5 sono suddivise in gruppi:

- Proprietà del Terminale Client - informazioni sul terminale client;
- Proprietà del programma-mql5 eseguite - proprietà dei programmi mql5, che aiutano a controllare la sua esecuzione;
- Proprietà del simbolo - ottiene informazioni su un simbolo;
- Proprietà account – informazioni sul corrente account;
- Statistiche Testing - risultati del testing dell' Expert Advisor.

Proprietà del Terminale Client

Le informazioni sul terminale client possono essere ottenute con due funzioni: TerminalInfoInteger() e TerminalInfoString(). Per i parametri, queste funzioni accettano valori ENUM_TERMINAL_INFO_INTEGER e ENUM_TERMINAL_INFO_STRING rispettivamente.

ENUM_TERMINAL_INFO_INTEGER

Identificatore	Descrizione	Tipo
TERMINAL_BUILD	Il numero di build del terminale client	int
TERMINAL_COMMUNITY_CONNECTION	Collegamento a MQL5.community	bool
TERMINAL_CONNECTED	La connessione ad un trade server	bool
TERMINAL_DLLS_ALLOWED	Il permesso di utilizzare DLL	bool
TERMINAL_TRADE_ALLOWED	Permesso per il trade	bool
TERMINAL_EMAIL_ENABLED	Il permesso di inviare e-mail utilizzando l' SMTP-server ed il login, specificato nelle impostazioni del terminale	bool
TERMINAL_FTP_ENABLED	Il permesso di inviare i reports con server FTP ed il login, specificato nelle impostazioni del terminale	bool
TERMINAL_NOTIFICATIONS_ENABLED	Permesso di inviare notifiche smartphone	bool
TERMINAL_MAXBARS	Il massimo conteggio di barre nel chart	int
TERMINAL_MQID	Il flag indica la presenza di dati ID MetaQuotes per le notifiche Push	bool
TERMINAL_CODEPAGE	Numero della pagina codice della lingua installata nel terminale client	int
TERMINAL_CPU_CORES	Il numero di CPU cores nel sistema	int
TERMINAL_DISK_SPACE	Spazio libero su disco per la cartella MQL5\Files del terminale(agente), MB	int
TERMINAL_MEMORY_PHYSICAL	Memoria fisica nel sistema, MB	int
TERMINAL_MEMORY_TOTAL	Memoria disponibile per il processo del terminale(agente), MB	int
TERMINAL_MEMORY_AVAILABLE	Memoria libera del processo del terminale(agente), MB	int

Identificatore	Descrizione	Tipo
TERMINAL_MEMORY_USED	Memoria utilizzata dal terminale(agente), MB	int
TERMINAL_X64	Indicazione del "terminale 64-bit"	bool
TERMINAL_OPENCL_SUPPORT	La versione del OpenCL supportato nel formato di 0x00010002 = 1.2. "0" significa che OpenCL non è supportato	int
TERMINAL_SCREEN_DPI	La risoluzione del display informativo sullo schermo è misurata come ammontare di Punti in una linea per Inch (DPI). Conoscendo il valore del parametro, puoi impostare la grandezza dell'oggetto grafico così che appaiono uguali su monitor con differenti caratteristiche di risoluzione.	int
TERMINAL_SCREEN_LEFT	La coordinata sinistra dello schermo virtuale. Uno schermo virtuale è un rettangolo che copre tutti i monitor. Se il sistema ha due monitor ordinati da destra a sinistra, la coordinata sinistra dello schermo virtuale può trovarsi sul bordo di due monitor.	int
TERMINAL_SCREEN_TOP	La coordinata superiore dello schermo virtuale	int
TERMINAL_SCREEN_WIDTH	Larghezza del terminale	int
TERMINAL_SCREEN_HEIGHT	Altezza del terminale	int
TERMINAL_LEFT	La coordinata sinistra del terminale relativa allo schermo virtuale	int
TERMINAL_TOP	La coordinata superiore del terminale relativa allo schermo virtuale	int
TERMINAL_RIGHT	La coordinata destra del terminale relativa allo schermo virtuale	int
TERMINAL_BOTTOM	La coordinata inferiore del terminale relativa allo schermo virtuale	int
TERMINAL_PING_LAST	L'ultimo valore ping conosciuto al trade server in millisecondi. Un	int

Identificatore	Descrizione	Tipo
	secondo comprende un milione di microsecondi.	
TERMINAL_VPS	Indicazione che il terminale è stato lanciato sul MetaTrader Virtual Hosting Server (MetaTrader VPS)	bool
L'identificatore Key	**Descrizione**	
TERMINAL_KEYSTATE_LEFT	Stato del tasto "Freccia Sinistra"	int
TERMINAL_KEYSTATE_UP	Stato del tasto "Freccia Su"	int
TERMINAL_KEYSTATE_RIGHT	Stato del tasto "Freccia Destra"	int
TERMINAL_KEYSTATE_DOWN	Stato del tasto "Freccia Giu"	int
TERMINAL_KEYSTATE_SHIFT	Stato del tasto "Shift"	int
TERMINAL_KEYSTATE_CONTROL	Stato del tasto "Ctrl"	int
TERMINAL_KEYSTATE_MENU	Stato del tasto "Windows"	int
TERMINAL_KEYSTATE_CAPSLOCK	Stato del tasto "CapsLock"	int
TERMINAL_KEYSTATE_NUMLOCK	Stato del tasto "BlocNum"	int
TERMINAL_KEYSTATE_SCRLOCK	Stato del tasto "ScrollLock"	int
TERMINAL_KEYSTATE_ENTER	Stato del tasto "Invio"	int
TERMINAL_KEYSTATE_INSERT	Stato del tasto "Ins"	int
TERMINAL_KEYSTATE_DELETE	Stato del tasto "Canc"	int
TERMINAL_KEYSTATE_HOME	Stato del tasto "Home"	int
TERMINAL_KEYSTATE_END	Stato del tasto "Fine"	int
TERMINAL_KEYSTATE_TAB	Stato del tasto "Tab"	int
TERMINAL_KEYSTATE_PAGEUP	Stato del tasto "PagSu"	int
TERMINAL_KEYSTATE_PAGEDOWN	Stato del tasto "PagGiu"	int
TERMINAL_KEYSTATE_ESCAPE	Stato del tasto "Esc"	int

La chiamata a TerminalInfoInteger(TERMINAL_KEYSTATE_XXX) restituisce lo stesso stato codice della chiave come GetKeyState() nella funzione in MSDN.

Esempio di calcolo del fattore di scala:

```
//--- Creazione di un bottone di spessore 1.5 sullo schermo
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // Trova il DPI del monito
int base_width = 144;                                      // Lo spessore di base in p
int width      = (button_width * screen_dpi) / 96;         // Calcola lo spessore del
```

```
...

//--- Calcolo del fattore di scala come percentuale
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- Uso del fattore di scala
width=(base_width * scale_factor) / 100;
width=(base_width * scale_factor) / 100;
```

Nell'esempio qui sopra, la risorsa grafica appare la stessa su monitor con caratteristiche di risoluzione differenti. La grandezza degli elementi di controllo (bottoni, finestre di dialogo, ecc.) corrisponde all'impostazione della personlizzazione.

ENUM_TERMINAL_INFO_DOUBLE

Identificatore	Descrizione	Tipo
TERMINAL_COMMUNITY_BALANCE	Bilancio in MQL5.community	double
TERMINAL_RETRANSMISSION	Percentuale di rispedizione dei pacchetti di rete nel protocollo TCP/IP per tutte le applicazioni e servizi in esecuzione sul computer specificato. La perdita di pacchetti si verifica anche nelle reti più veloci e configurate correttamente. In questo caso, non viene confermata la consegna del pacchetto tra il destinatario ed il mittente, pertanto i pacchetti persi vengono nuovamente inviati. Non è un'indicazione della qualità della connessione tra un particolare terminale ed un trade server, poiché la percentuale viene calcolata per l'intera attività di rete, inclusa l'attività di sistema e di backgroud. Il valore TERMINAL_RETRANSMISSION viene richiesto dal sistema operativo una volta al minuto. Il terminale in sè non calcola questo valore.	double

Le operazioni sui file possono essere eseguite solo in due directory; i percorsi corrispondenti possono essere ottenuti utilizzando la richiesta per le proprietà TERMINAL_DATA_PATH e TERMINAL_COMMONDATA_PATH.

ENUM_TERMINAL_INFO_STRING

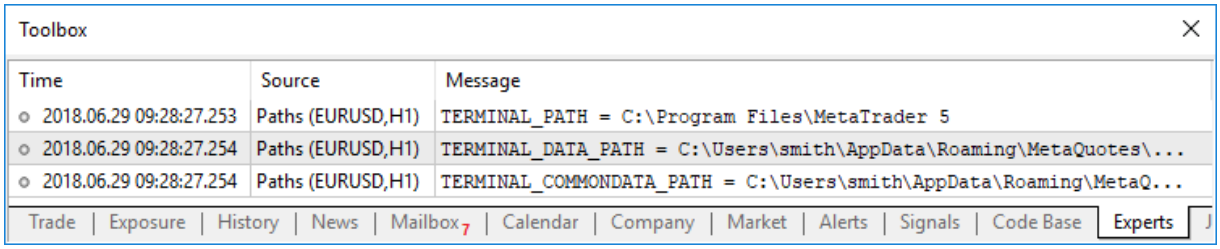
Identificatore	Descrizione	Tipo
TERMINAL_LANGUAGE	Lingua del terminale	string
TERMINAL_COMMUNITY_ACCOUNT	Account in MQL5.community	string
TERMINAL_COMPANY	Nome della società	string
TERMINAL_NAME	Nome del terminale	string
TERMINAL_PATH	Cartella da cui viene avviato il terminale	string
TERMINAL_DATA_PATH	Cartella in cui sono memorizzati i dati dei terminali	string
TERMINAL_COMMONDATA_PATH	Percorso comune per tutti i terminali installati in un computer	string
TERMINAL_CPU_NAME	Nome CPU	string
TERMINAL_CPU_ARCHITECTURE	Architettura CPU	string
TERMINAL_OS_VERSION	Nome del Sistema Operativo dell'utente	string

Per una migliore comprensione dei percorsi memorizzati nelle proprietà di TERMINAL_PATH, TERMINAL_DATA_PATH e parametri TERMINAL_COMMONDATA_PATH, si consiglia di eseguire lo script, che restituirà questi valori per la copia corrente del terminale client, installato sul computer

Esempio: Lo script restituisce le informazioni riguardo i percorsi del terminale client.

```
//+------------------------------------------------------------------------------+
//|                                                  Check_TerminalPaths.mq5 |
//|                                Copyright 2009, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

Come risultato dell'esecuzione dello script nel Journal degli Experts si vedrà un messaggio, come il seguente:



Proprietà dell' Esecuzione di Programmi MQL5

Per ottenere informazioni sul programma MQL5 attualmente in esecuzione, vengono utilizzate le costanti di ENUM_MQL_INFO_INTEGER e ENUM_MQL_INFO_STRING.

Per la funzione MQLInfoInteger

ENUM_MQL_INFO_INTEGER

Identificatore	Descrizione	Tipo
MQL_HANDLES_USED	The current number of active object handles. These include both dynamic (created via new) and non-dynamic objects, global/local variables or class members. The more handles a program uses, the more resources it consumes.	int
MQL_MEMORY_LIMIT	Eventuale quantità massima di memoria dinamica per MQL5 programma in MB	int
MQL_MEMORY_USED	La grandezza della memoria utilizzata dal MQL5 programma in MB	int
MQL_PROGRAM_TYPE	Tipo di programma MQL5	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	Il permesso di utilizzare DLL per un determinato programma eseguito	bool
MQL_TRADE_ALLOWED	Il permesso al trade per un determinato	bool

Identificatore	Descrizione	Tipo
	programma eseguito	
MQL_SIGNALS_ALLOWED	Il permesso di modificare i Segnali **per un determinato programma eseguito**	bool
MQL_DEBUG	Indicazione che il programma sta girando in modalità debugging	bool
MQL_PROFILER	Indicazione che il programma sta girando in modalità profiling del codice	bool
MQL_TESTER	Indicazione che il programma sta girando nel tester	bool
MQL_FORWARD	Indication that the program is running in the forward testing process	bool
MQL_OPTIMIZATION	Indicazione che il programma sta girando in modalità ottimizzazione	bool
MQL_VISUAL_MODE	Indicazione che il programma sta girando in modalità testing visuale	bool
MQL_FRAME_MODE	Indicazione che l'Expert advisor sta girando in in modalità raccolta frames dei risultati di ottimizzazione	bool

Identificatore	Descrizione	Tipo
MQL_LICENSE_TYPE	Tipo di licenza del modulo EX5. La licenza si riferisce al modulo EX5, da cui viene effettuata una richiesta utilizzando MQLInfoInteger (MQL_LICENSE_TYPE).	ENUM_LICENSE_TYPE

Per la funzioneMQLInfoString

ENUM_MQL_INFO_STRING

Identificatore	Descrizione	Tipo
MQL_PROGRAM_NAME	Nome del programma MQL5 eseguito	string
MQL5_PROGRAM_PATH	Percorso **per il programma eseguito dato**	string

Per informazioni sul tipo di programma in esecuzione, vengono utilizzati i valori di ENUM_PROGRAM_TYPE.

ENUM_PROGRAM_TYPE

Identificatore	Descrizione
PROGRAM_SCRIPT	Script
PROGRAM_EXPERT	Expert
PROGRAM_INDICATOR	Indicatore
PROGRAM_SERVICE	Servizio

ENUM_LICENSE_TYPE

Identificatore	Descrizione
LICENSE_FREE	Una versione gratuita ed illimitata

Identificatore	Descrizione
LICENSE_DEMO	Una versione di prova di un prodotto dal Market. Funziona solo nello strategy tester
LICENSE_FULL	Una versione di licenza acquistata che permette almeno 5 attivazioni. Il venditore può incrementare il numero consentito di attivazioni
LICENSE_TIME	Una versione con termini di licenza limitati

Esempio:

```
   ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQLInfoInteger(MQL_PROGRAM_TYPE);
   switch(mql_program)
     {
      case PROGRAM_SCRIPT:
        {
         Print(__FILE__+" is script");
         break;
        }
      case PROGRAM_EXPERT:
        {
         Print(__FILE__+" is Expert Advisor");
         break;
        }
      case PROGRAM_INDICATOR:
        {
         Print(__FILE__+" is custom indicator");
         break;
        }
      default:Print("MQL5 type value is ",mql_program);
   //---
   Print("MQLInfoInteger(MQL_MEMORY_LIMIT)=", MQLInfoInteger(MQL_MEMORY_LIMIT), " MB")
   Print("MQLInfoInteger(MQL_MEMORY_USED)=", MQLInfoInteger(MQL_MEMORY_USED), " MB");
   Print("MQLInfoInteger(MQL_HANDLES_USED)=", MQLInfoInteger(MQL_HANDLES_USED), " hanc
```

Proprietà dei Simboli

Per ottenere le correnti informazioni di mercato ci sono diverse funzioni: SymbolInfoInteger(), SymbolInfoDouble() e SymbolInfoString(). Il primo parametro è il nome del simbolo, i valori del parametro della seconda funzione possono essere uno degli identificatori di ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE e ENUM_SYMBOL_INFO_STRING.

Per la funzioneSymbolInfoInteger()

ENUM_SYMBOL_INFO_INTEGER

Identificatore	Descrizione	Tipo
SYMBOL_SECTOR	The sector of the economy to which the asset belongs	ENUM_SYMBOL_SECTOR
SYMBOL_INDUSTRY	The industry or the economy branch to which the symbol belongs	ENUM_SYMBOL_INDUSTRY
SYMBOL_CUSTOM	È un sim	bool

Identificatore	Descrizione	Tipo
	bolo personalizzato - il simbolo è stato creato in modo sintetico basato su altri simboli dal Market Watch e/o da origini dati esterne	
SYMBOL_BACKGROUND_COLOR	Il colore dello sfondo utilizzato	color

Identificatore	Descrizione	Tipo
	per il simbolo nel Market Watch	
SYMBOL_CHART_MODE	Il tipo di prezzo usato per generare barre dei simboli, vale a dire Bid o Last	ENUM_SYMBOL_CHART_MODE
SYMBOL_EXIST	Il simbolo con questo nome esiste	bool
SYMBOL_SELECT	Symbol è sele	bool

Identificatore	Descrizione	Tipo
	zionato in MarketWatch	
SYMBOL_VISIBLE	Il simbolo è visibile in MarketWatch. Alcuni simboli (per lo più, questi sono i tassi di transazione (cross rates) necessari per il calcolo dei	bool

Identificatore	Descrizione	Tipo
	requisiti di margine o profitti in valuta di deposito) vengono selezionati automaticamente, ma in genere non sono visibili nel MarketWatch. Per essere visualizzati, tali	

Identificatore	Descrizione	Tipo
	simboli devono essere selezionati esplicitamente.	
SYMBOL_SESSION_DEALS	Numero di affari nella sessione corrente	long
SYMBOL_SESSION_BUY_ORDERS	Numero di ordini Buy in questo momento	long
SYMBOL_SESSION_SELL_ORDERS	Numero di ordini Sell in questo	long

Identificatore	Descrizione	Tipo
	momento	
SYMBOL_VOLUME	Volume dell'ultima transazione	long
SYMBOL_VOLUMEHIGH	Volume massimo del giorno	long
SYMBOL_VOLUMELOW	Volume minimo del giorno	long
SYMBOL_TIME	Orario dell'ultima quotazione	datetime
SYMBOL_TIME_MSC	Tempo dell'ultima quotazione in milli	long

Identificatore	Descrizione	Tipo
	secondi dal 1970.01.01	
SYMBOL_DIGITS	Cifre dopo la virgola decimale	int
SYMBOL_SPREAD_FLOAT	Indicazione di uno spread fluttuante	bool
SYMBOL_SPREAD	Valore differenziale in punti	int
SYMBOL_TICKS_BOOKDEPTH	Numero massimo di richieste mostrate in Profondi	int

Identificatore	Descrizione	Tipo
	tà di Mercato. Per i simboli che non hanno coda di richieste, il valore è uguale a zero.	
SYMBOL_TRADE_CALC_MODE	Modalità di calcolo del prezzo del contratto	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	Tipo ordine di esecuzione	ENUM_SYMBOL_TRADE_MODE

Identificatore	Descrizione	Tipo
SYMBOL_START_TIME	Data di inizio simbolo di trade (di solito usato per i futures)	datetime
SYMBOL_EXPIRATION_TIME	Data di scadenza del commercio di trade (di solito usato per i futures)	datetime
SYMBOL_TRADE_STOPS_LEVEL	Rientro minimo in punti dal	int

Identificatore	Descrizione	Tipo
	prezzo corrente più vicino per piazzare ordini di Stop	
SYMBOL_TRADE_FREEZE_LEVEL	Distanza per congelare le operazioni di trade in punti	int
SYMBOL_TRADE_EXEMODE	Modalità di esecuzione affare	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	Modalità di calcolo swap	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	Giorno dell	ENUM_DAY_OF_WEEK

Identificatore	Descrizione	Tipo
	a settimana per caricare swap di rollover a 3 giorni	
SYMBOL_MARGIN_HEDGED_USE_LEG	Calcolo del margine di copertura utilizzando la gamba più grande (Buy o Sell)	bool
SYMBOL_EXPIRATION_MODE	Flags di ordine consentito modalità di espi	int

Identificatore	Descrizione	Tipo
	razione	
SYMBOL_FILLING_MODE	Flags di ordine consentito modalità di riempimento	int
SYMBOL_ORDER_MODE	Flags dei consentiti tipi di ordine	int
SYMBOL_ORDER_GTC_MODE	Scadenza degli ordini Stop Loss o Take Profit, se SYMBOL_EXPIR	ENUM_SYMBOL_ORDER_GTC_MODE

Identificatore	Descrizione	Tipo
	ATION_MODE=SYMBOL_EXPIRATION_GTC (Buono fino ad annullamento)	
SYMBOL_OPTION_MODE	Tipo di Opzione	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	Opzione destra (Chiamata/Mesa)	ENUM_SYMBOL_OPTION_RIGHT

Per la funzione SymbolInfoDouble()

ENUM_SYMBOL_INFO_DOUBLE

Identificatore	Descrizione	Tipo
SYMBOL_BID	Bid - migliore offerta sell	double
SYMBOL_BIDHIGH	Massimo Bid	double

Identificatore	Descrizione	Tipo
	del giorno	
SYMBOL_BIDLOW	Minimo Bid del giorno	double
SYMBOL_ASK	Ask - migliore offerta buy	double
SYMBOL_ASKHIGH	Massimo Ask del giorno	double
SYMBOL_ASKLOW	Minimo Ask del giorno	double
SYMBOL_LAST	Prezzo dell'ultima transazione	double
SYMBOL_LASTHIGH	Ultima massima del giorno	double
SYMBOL_LASTLOW	Ultima minima del giorno	double
SYMBOL_VOLUME_REAL	Volume dell'ultima transazione	double
SYMBOL_VOLUMEHIGH_REAL	Volume massimo del giorno	double
SYMBOL_VOLUMELOW_REAL	Volume minimo del giorno	double

Identificatore	Descrizione	Tipo
SYMBOL_OPTION_STRIKE	Il prezzo di strike di un'opzione. Il prezzo al quale un acquirente di opzioni può acquistare (in un'opzione Call) o vendere (in un'opzione Put) l'attività sottostante, ed il venditore di opzione è obbligato a vendere o acquistare l'importo corrispondente dell'attività sottostante.	double
SYMBOL_POINT	Valore punti	double

Identificatore	Descrizione	Tipo
	del Symbol	
SYMBOL_TRADE_TICK_VALUE	Valore di SYMBOL_TRADE_TICK_VALUE_PROFIT	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	Prezzo tick calcolato per una posizione favorevole	double
SYMBOL_TRADE_TICK_VALUE_LOSS	Prezzo tick calcolato per una posizione in perdita	double
SYMBOL_TRADE_TICK_SIZE	Cambiamento prezzo minimo	double
SYMBOL_TRADE_CONTRACT_SIZE	Grandezza del contratto di trade	double
SYMBOL_TRADE_ACCRUED_INTEREST	Interessi maturati - interessi cedola accumulati, vale a dire	double

Identificatore	Descrizione	Tipo
	parte dell'interesse cedola calcolato in proporzione al numero di giorni successivi all'emissione coupon o all'ultimo pagamento di interessi coupon	
SYMBOL_TRADE_FACE_VALUE	Valore nominale - valore iniziale di obbligazione impostato dall'emittente	double
SYMBOL_TRADE_LIQUIDITY_RATE	La liquidità è la quota dell'attività che può essere utilizzata per il	double

Identificatore	Descrizione	Tipo
	margine.	
SYMBOL_VOLUME_MIN	Volume minimo per un affare	double
SYMBOL_VOLUME_MAX	Volume massimo per un affare	double
SYMBOL_VOLUME_STEP	Cambio di step minimo per l'esecuzione dell'affare	double
SYMBOL_VOLUME_LIMIT	Massimo volume consentito aggregato di una posizione aperta ed ordini pendenti in una direzione (acquisto o vendita) per il simbolo. Per esempio, con la limitazione di 5	double

Identificatore	Descrizione	Tipo
	lotti, si può avere una posizione aperta buy con il volume di 5 lotti e posizionare un ordine pendente Sell Limit con il volume di 5 lotti. Ma in questo caso non è possibile inserire un ordine pendente Buy Limit (in quanto il volume totale in una direzione supererà il limite) o piazzare Sell	

Identificatore	Descrizione	Tipo
	Limit con il volume più di 5 lotti.	
SYMBOL_SWAP_LONG	Valore di swap Long	double
SYMBOL_SWAP_SHORT	Valore di swap Short	double
SYMBOL_SWAP_SUNDAY	Rapporto di calcolo dello swap (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) per le posizioni overnight rinnovate da DOMENICA al giorno successivo. Sono supportati i seguenti valori: • 0 - nonv	double

Identificatore	Descrizione	Tipo
	ienea ddebi tato alcu noswa p • 1 - swap sing olo • 3 - swap trip	

Identificatore	Descrizione	Tipo
	lo	
SYMBOL_SWAP_MONDAY	Rapporto di calcolo dello swap (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) per posizioni overnight rinnovate dal lunedì al martedì	double
SYMBOL_SWAP_TUESDAY	Rapporto di calcolo dello swap (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) per posizioni overnight rinnovate da martedì a mercoledì	double

Identificatore	Descrizione	Tipo
SYMBOL_SWAP_WEDNESDAY	Rapporto di calcolo dello swap (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) per posizioni overnight rinnovate da mercoledì a giovedì	double
SYMBOL_SWAP_THURSDAY	Rapporto di calcolo dello swap (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) per posizioni overnight rinnovate da giovedì a venerdì	double
SYMBOL_SWAP_FRIDAY	Rapporto di calcolo	double

Identificatore	Descrizione	Tipo
	dello swap (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) per posizioni overnight rinnovate da venerdì a sabato	
SYMBOL_SWAP_SATURDAY	Rapporto di calcolo dello swap (SYMBOL_SWAP_LONG o SYMBOL_SWAP_SHORT) per posizioni overnight rinnovate da sabato a domenica	double
SYMBOL_MARGIN_INITIAL	Il margine iniziale indica l'import	double

Identificatore	Descrizione	Tipo
	o nella valuta di margine richiesto per aprire una posizione con il volume di un lotto. E' usato per controllare le attività di un cliente quando lui o lei entra nel mercato. La funzione SymbolInfoMarginRate() fornisce dati sulla quantità di margine addebitato in base al tipo di ordine ed alla direzione.	

Identificatore	Descrizione	Tipo
SYMBOL_MARGIN_MAINTENANCE	Il margine di mantenimento. Se è impostato, imposta la quantità margine nella valuta margine del simbolo, caricato di un lotto. E' usato per controllare le attività di un cliente quando cambiano gli status del suo account. Se il margine di mantenimento è uguale a 0, viene utilizzato il margine iniziale.	double

Identificatore	Descrizione	Tipo
	La funzione SymbolInfoMarginRate() fornisce dati sulla quantità di margine addebitato in base al tipo di ordine ed alla direzione.	
SYMBOL_SESSION_VOLUME	Sommatoria del volume degli affari della sessione corrente	double
SYMBOL_SESSION_TURNOVER	Sommatoria del turnover della sessione corrente	double
SYMBOL_SESSION_INTEREST	Sommatoria dell'interesse aperto	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Volume corrente di	double

Identificatore	Descrizione	Tipo
	ordini Buy	
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Corrente volume di ordini Sell	double
SYMBOL_SESSION_OPEN	Prezzo di apertura della sessione corrente	double
SYMBOL_SESSION_CLOSE	Prezzo di chiusura della sessione corrente	double
SYMBOL_SESSION_AW	Prezzo medio ponderato della sessione corrente	double
SYMBOL_SESSION_PRICE_SETTLEMENT	Prezzo di liquidazione della sessione corrente	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	Prezzo minimo della session	double

Identificatore	Descrizione	Tipo
	e corrente	
SYMBOL_SESSION_PRICE_LIMIT_MAX	Prezzo massimo della sessione corrente	double
SYMBOL_MARGIN_HEDGED	Dimensione del contratto o valore di margine per un lotto di posizioni hedged (posizioni opposte di un simbolo). Sono possibili due metodi di calcolo dei margini per le posizioni hedged. Il metodo di calcolo è definito dal broker.	double

Identificatore	Descrizione	Tipo
	Calcolo base: • Se il margine iniziale (SYMBOL_MARGIN_INITIAL) è specificato per un simbolo, il margine coperto(hedged) viene specificato come valore assoluto (in termini monetari). • Se il margine iniziale non è specificato (uguale a	

Identificatore	Descrizione	Tipo
	0), SYMBOL_MARGIN_HEDGED è uguale alla dimensione del contratto che verrà utilizzato per calcolare il margine dalla formula appropriata in base al tipo dello strumento finanziario (SYMBOL_TRADE_CALC_MODE). Calcolo per la posizion	

Identificatore	Descrizione	Tipo
	e più grande: • Il valore SYMBOL_MARGIN_HEDGED non viene preso in considerazione. • Viene calcolato il volume di tutte le posizioni short e long di un simbolo. • Per ciascuna direzione viene calcolato un prezzo medio ponderato aperto ed	

Identificatore	Descrizione	Tipo
	un tasso medio di conversione ponderato alla valuta di deposito. • Successivamente, utilizzando la formula appropriata scelta in base al tipo di simbolo (SYMBOL_TRADE_CALC_MODE) il margine è calcolato per la parte short e la	

Identificatore	Descrizione	Tipo
	parte long. • Il più grande dei valori viene utilizzato come margine.	
SYMBOL_PRICE_CHANGE	Variazione del prezzo corrente rispetto alla fine del giorno di negoziazione precedente in %	double
SYMBOL_PRICE_VOLATILITY	Volatilità dei prezzi in %	double
SYMBOL_PRICE_THEORETICAL	Prezzo teorico dell'opzione	double
SYMBOL_PRICE_DELTA	Il delta dell' opzione/warrant mostra il valore in base al quale il prezzo dell'opzi	double

Identificatore	Descrizione	Tipo
	one cambia, quando il prezzo dell'attività sottostante (underlying asset) cambia di 1	
SYMBOL_PRICE_THETA	Il theta dell' opzione /warrant mostra il numero di punti che il prezzo dell'opzione deve perdere ogni giorno a causa di una rottura temporanea, ovvero quando si avvicina la data di scadenza	double
SYMBOL_PRICE_GAMMA	Il gamma dell'opzi	double

Identificatore	Descrizione	Tipo
	one/warrant mostra il tasso di variazione del delta - quanto velocemente o quanto lentamente cambia il premio dell'opzione	
SYMBOL_PRICE_VEGA	Il vega dell'opzione/warrant mostra il numero di punti in cui il prezzo dell'opzione cambia quando la volatilità cambia dell' 1%	double
SYMBOL_PRICE_RHO	Il rho dell' opzione /warrant riflette la sensibilità del	double

Identificatore	Descrizione	Tipo
	prezzo teorico dell'opzione rispetto al tasso di interesse che varia dell'1%	
SYMBOL_PRICE_OMEGA	Omega dell'opzione/warrant. L'elasticità dell'opzione mostra una variazione percentuale relativa del prezzo dell'opzione in base alla variazione percentuale del prezzo dell'attività sottostante	double
SYMBOL_PRICE_SENSITIVITY	La sensibilità dell'opzione/warrant	double

Identificatore	Descrizione	Tipo
	mostra di quanti punti il prezzo dell'attività sottostante (underlying asset) dell'opzione dovrebbe cambiare, in modo che il prezzo dell'opzione cambi di un punto	

Per le funzioni SymbolInfoString()

ENUM_SYMBOL_INFO_STRING

Identificatore	Descrizione	Tipo
SYMBOL_BASIS	L'asset sottostante di un derivato	string
SYMBOL_CATEGORY	Il nome del settore o categoria al quale appartiene il simbolo di trading	string

Identificatore	Descrizione	Tipo
SYMBOL_COUNTRY	The country to which the financial symbol belongs	string
SYMBOL_SECTOR_NAME	The sector of the economy to which the financial symbol belongs	string
SYMBOL_INDUSTRY_NAME	The industry branch or the industry to which the financial symbol belongs	string
SYMBOL_CURRENCY_BASE	Valuta base di un simbolo	string
SYMBOL_CURRENCY_PROFIT	Valuta di profitto	string
SYMBOL_CURRENCY_MARGIN	Valuta di margine	string
SYMBOL_BANK	Feeder della corrente quotazione	string
SYMBOL_DESCRIPTION	Descrizione del simbolo	string
SYMBOL_EXCHANGE	Il nome dell' exchange	string

Identificatore	Descrizione	Tipo
	in cui si fa trading del simbolo finanziario	
SYMBOL_FORMULA	La formula utilizzata per il prezzo personalizzato dei simboli. Se il nome di un simbolo finanziario utilizzato nella formula inizia con una cifra o contiene un carattere speciale (">" " ", "." "-", "&", "#" e cosi via.) devono essere utilizzate le virgolette attorno a questo nome simbolo. • Simbol	string

Identificatore	Descrizione	Tipo
	o sintetico: "@ESU19" / EURCAD • Calendar spread: "Si-9.1	

Identificatore	Descrizione	Tipo
	3"-"Si-6.13" • Euroindex:34.38805726*pow(EURUSD,0.3	

Identificatore	Descrizione	Tipo
	155)*pow(EURGBP,0.30056)*pow(EURJPY,0.18911)*pow(E	

Identificatore	Descrizione	Tipo
	URCHF,0.11113)*pow(EURSEK,0.07855)	
SYMBOL_ISIN	Il nome di un simbolo nel sistema ISIN (International Securities Identification Number). L'International Securities	string

Identificatore	Descrizione	Tipo
	Identification Number è un codice a 12 cifre alfanumerico che identifica in modo univoco una security. La presenza di questa proprietà del simbolo è determinata dal lato del trade server.	
SYMBOL_PAGE	L'indirizzo della pagina web contenente informazioni sui simboli. Questo indirizzo verrà visualizzato come un collegamento durante la visualizzazione delle proprietà dei simboli nel terminale	string

Identificatore	Descrizione	Tipo
SYMBOL_PATH	Percorso nell'albero del simbolo	string

Un chart dei prezzi dei simboli può essere basato su prezzi Bid o Last. Il prezzo scelto per i chart dei simboli influenza anche la generazione e la visualizzazione di barre nel terminale. I valori possibili della proprietà SYMBOL_CHART_MODE sono descritti in ENUM_SYMBOL_CHART_MODE

ENUM_SYMBOL_CHART_MODE

Identificatore	Descrizione
SYMBOL_CHART_MODE_BID	Le barre si basano sui prezzi Bid
SYMBOL_CHART_MODE_LAST	Le barre si basano sui prezzi Last

Per ogni simbolo, possono essere specificate diverse modalità di espirazione di ordini pendenti. Una flag viene è abbinata a ciascuna modalità. Le flags possono essere combinati utilizzando l'operazione di logica**OR** (||), per esempio, SYMBOL_EXPIRATION_GTC|SYMBOL_EXPIRATION_SPECIFIED. Al fine di verificare se una determinata modalità è consentita per il simbolo, il risultato della logica **AND** (&) dovrebbe essere confrontato con la modalità flag.

Se è specificato il flag SYMBOL_EXPIRATION_SPECIFIED per un simbolo, allora durante l'invio di un ordine pendente, è possibile specificare il momento fino a che che questo ordine pendente sarà valido.

Identificatore	Valore	Descrizione
SYMBOL_EXPIRATION_GTC	1	L'ordine è valido durante un periodo di tempo illimitato, fino a quando non viene esplicitamente annullato
SYMBOL_EXPIRATION_DAY	2	L'ordine è valido fino alla fine della giornata
SYMBOL_EXPIRATION_SPECIFIED	4	L'orario di espirazione viene specificato nell'ordine
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	L'orario di scadenza è specificato nell'ordine

Esempio:

```
//+------------------------------------------------------------------------------+
//| Controlla se la modalità di espirazione specificata è consentita             |
//+------------------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
```

```
  {
//--- Ottiene il valore della proprietà che descrive le modalità di espirazione ammess
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- Restituisce true, se la modalità exp_type è consentita
   return((expiration&exp_type)==exp_type);
  }
```

Se la proprietà SYMBOL_EXPIRATION_MODE è impostata su SYMBOL_EXPIRATION_GTC (buono fino alla cancellazione), la scadenza degli ordini pendenti, nonché degli ordini di Stop Loss/Take Profit dovrebbe essere impostata ulteriormente utilizzando l'enumerazione ENUM_SYMBOL_ORDER_GTC_MODE.

ENUM_SYMBOL_ORDER_GTC_MODE

Identificatore	Descrizione
SYMBOL_ORDERS_GTC	Gli ordini pendenti e i livelli di Stop Loss/Take Profit sono validi per un periodo illimitato fino alla loro cancellazione esplicita
SYMBOL_ORDERS_DAILY	Gli ordini sono validi durante un giorno di trading. Alla fine della giornata vengono eliminati tutti i livelli di Stop Loss e Take Profit, nonché gli ordini pendenti.
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	Quando cambia un giorno trade, vengono eliminati solo gli ordini pendenti, mentre i livelli di Stop Loss e Take Profit vengono mantenuti.

Quando si invia un ordine, è possibile specificare il criterio di riempimento per il volume impostato nel l'ordine. Le modalità di riempimento ordine ammesse per ogni simbolo sono indicate nella tabella. È possibile impostare diverse modalità di un simbolo dalla combinazione di flags. I flag possono essere combinati dal funzionamento della logica **OR** (|), ad esempio,SYMBOL_FILLING_FOK | SYMBOL_FILLING_IOC. Al fine di verificare se una determinata modalità è consentita per il simbolo, il risultato dell' **AND logico** (&) dovrebbe essere confrontato con la modalità flag.

Fill Policy	Identificatore	Valore	Descrizione
Fill or Kill	SYMBOL_FILLING_FOK	1	Questa regola significa che un affar

Fill Policy	Identificatore	Valore	Descrizione
			e può essere eseguito solo con il volume specificato. Se la quantità necessaria di uno strumento finanziario non è attualmente disponibile sul mercato, l'ordine non verrà eseguito. Il volume richiesto può essere riempito con diver

Fill Policy	Identificatore	Valore	Descrizione
			se offerte disponibili sul mercato al momento.
Immediate or Cancel	SYMBOL_FILLING_IOC	2	In questo caso un trader si impegna ad eseguire un affare con il volume massimo disponibile sul mercato entro quello indicato nell'ordine. Nel caso in cui l'ordine non può esser

Fill Policy	Identificatore	Valore	Descrizione
			e riempito completamente, il volume disponibile dell'ordine sarà riempito, e il volume rimanente verrà annullato. La possibilità di utilizzare gli ordini IOC è determinata dal trade server.
Passivo	SYMBOL_FILLING_BOC	4	La politica BcC (Book-or-Cancel)

Fill Policy	Identificatore	Valore	Descrizione
			presuppone che un ordine possa essere effettuato solo nel Depth of Market e non possa essere eseguito immediatamente. Se l'ordine può essere eseguito immediatamente quando piazzato, allor

Fill Policy	Identificatore	Valore	Descrizione
			a viene annullato. Infatti, questa politica di esecuzione può essere specificata solo quando il prezzo dell'ordine piazzato è peggiore del mercato attuale. Gli ordini BoC sono utilizzati per implementare

Fill Policy	Identificatore	Valore	Descrizione
			la negoziazione passiva, in modo che l'ordine non venga eseguito immediatamente al momento dell'immissione e non influisca sulla liquidità corrente. Sono supportati solo gli ordini limit e stop limit

Fill Policy	Identificatore	Valore	Descrizione
			, ovvero la flag SYMBOL_ORDER_MODE dovrebbe contenere i valori SYMBOL_ORDER_LIMIT e/o SYMBOL_ORDER_STOP_LIMIT.
Return	Nessun Identificatore		Questa policy viene utilizzata solo per ordini di mercato (Buy e Sell), ordini limit

Fill Policy	Identificatore	Valore	Descrizione
			e stop limit e solo per i simboli con l'esecuzione a Mercato o Exchange. In caso di riempimento parziale di un ordine di mercato o di ordine limit con un volume rimanente, non viene cancellato, ma ulteriormente elaborato.

Nella modalità di esecuzione a Richiesta ed Istantanea, per gli ordini di mercato viene sempre usata la policy di riempimento Fill or Kill, e la policy Return viene sempra usata per gli ordini limit. In questo caso, quando si inviano ordini usando OrderSend o OrderSendAsync, non è necessario specificare un criterio di riempimento per essi.

Nelle modalità di esecuzione Market ed Exchange la polocy Return è sempre consentito per tutti i tipi di ordine. Per scoprire se le altre policy sono ammesse, utilizzare le proprietà SYMBOL_FILLING_FOK e SYMBOL_FILLING_IOC.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Controlla se la modalità di riempimento specificata è consentita             |
//+------------------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- Ottiene il valore della proprietà che descrive le modalità di riempimento conser
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- Restituisce true, se la modalità fill_type è consentita
   return((filling & fill_type)==fill_type);
  }
```

Quando si invia una richiesta di trade usando la funzione OrderSend() , un tipo di ordine dall'enumerazione ENUM_ORDER_TYPE deve essere specificato per alcune operazioni. Non tutti i tipi di ordini possono essere consentiti per uno specifico simbolo. SYMBOL_ORDER_MODE la proprietà descrive le flags dei tipi di ordine consentiti.

Identificatore	Valore	Descrizione
SYMBOL_ORDER_MARKET	1	Gli ordini di Mercato sono consentiti (Buy e Sell)
SYMBOL_ORDER_LIMIT	2	Gli ordini Limit sono ammessi (Buy Limit e Sell Limit)
SYMBOL_ORDER_STOP	4	Ordini di Stop sono ammessi (Buy Stop e Sell Stop)
SYMBOL_ORDER_STOP_LIMIT	8	Ordini Stop-limit sono ammessi (Buy Stop Limit e Sell Stop Limit)
SYMBOL_ORDER_SL	16	Stop Loss è consentito
SYMBOL_ORDER_TP	32	Take Profit è consentito
SYMBOL_ORDER_CLOSEBY	64	Autorizzazione di una Chiusura Per Operazione, ovvero chiusura di una posizione da un'altra posizione aperta sugli stessi strumenti, ma nella direzione opposta. La proprietà è impostata per gli account con il sistema di account

Identificatore	Valore	Descrizione
		hedging (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)

Esempio:

```
//+------------------------------------------------------------------------------+
//| La funzione stampa i tipi di ordine consentiti per il simoblo                |
//+------------------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- riceve il valore della proprietà che descrivendo i tipi di ordine consentiti
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- controlla ordini di mercato (Market Execution)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Gli ordini di Mercato sono consentiti (Buy e Sell)");
//--- controlla Ordini Limit
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Ordini Buy Limit e Sell Limit sono consentiti");
//--- controlla Ordini di Stop
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Ordini Buy Stop e Sell Stop sono consentiti");
//--- controlla ordini Stop Limit
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Ordini Buy Stop Limit e Sell Stop Limit sono consentiti");
//--- controlla se il piazzare ordini Stop Loss è consentito
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+":Ordini Stop Loss sono consentiti");
//--- controlla se piazzare ordini Take Profit è consentito
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+":Ordini Take Profit sono consentiti");
//---
  }
```

L'enumerazione ENUM_SYMBOL_CALC_MODE viene utilizzata per ottenere informazioni su come i requisiti di margine per un simbolo vengono calcolati.

ENUM_SYMBOL_CALC_MODE

Identificatore	Descrizione	Formula
SYMBOL_CALC_MODE_FOREX	Modalità Forex -	Margine: Lotti*Grandezza_Contratto/Leva * Tasso_del_Margine

Identificatore	Descrizione	Formula
	calcolo del margine e del di profitto per Forex	Profitto: (prezzo_close - prezzo_open) * Grandezza_contratto * Lotti
SYMBOL_CALC_MODE_FOREX_NO_LEVERAGE	Forex Modalità No Leva – calcolo del profitto e margine per i simboli Forex senza tenere in considerazione la leva	Margine: Lotti * Grandezza_Contratto * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * Grandezza_Contratto * Lotti
SYMBOL_CALC_MODE_FUTURES	Modalità Futures - calcolo del margine e	Margine: Lotti * MargineIniziale * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * PrezzoTick / GrandezzaTick * Lotti

Identificatore	Descrizione	Formula
	del profitto per i futures	
SYMBOL_CALC_MODE_CFD	Modalità CFD - calcolo del margine e del profitto e CFD	Margine: Lotti * Grandezza * PrezzoDiMercato * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * Grandezza_Contratto * Lotti
SYMBOL_CALC_MODE_CFDINDEX	Modalità indici CFD - calcolo del margine e del profitto per indici CFD	Margine: (Lotti * GrandezzaContratto * PrezzoDiMercato) / PrezzoTick / GrandezzaTick * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * Grandezza_Contratto * Lotti
SYMBOL_CALC_MODE_CFDLEVERAGE	Modalità CFD Leverage - calcolo del margine e del profi	Margine: (Lotti * GrandezzaContratto * PrezzoDiMercato) / Leva * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * Grandezza_Contratto * Lotti

Identificatore	Descrizione	Formula
	tto per CFD al trading leverage	
SYMBOL_CALC_MODE_EXCH_STOCKS	Modalità di Scambio - calcolo del margine e del profitto per i titoli di trade in uno stock exchange / borsa	Margine: Lotti * GrandezzaContratto * LastPrice * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * Grandezza_Contratto * Lotti
SYMBOL_CALC_MODE_EXCH_FUTURES	Modalità Futures - calcolo del margine e del profitto per i contratti	Margine: Lotti * MargineIniziale * Margin_Rate o Lotti * MargineDiMantenimento * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * Lotti * PrezzoTick / GrandezzaTick

Identificatore	Descrizione	Formula
	futures in uno stock exchange / borsa	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	Modalità FORTS Futures - calcolo del margine e del profitto per i contratti futures sui FORTS. Il margine può essere ridotto dalla quantità di deviazione MarginDiscoun	Margine: Lotti * MargineIniziale * Tasso_del_Margine o Lotti * Margine_Di_Mantenimento * Tasso_del_Margine Profitto: (prezzo_close - prezzo_open) * Lotti * PrezzoTick / GrandezzaTick

Identificatore	Descrizione	Formula
	t secondo le seguenti regole: 1. Se il prezzo di una posizione long (ordine di acquisto buy) è inferiore al prezzo stimato, MarginDiscount = Lotti * ((PriceSettle-PrezzoOrdine) * PrezzoTick / Grandezz	

Identificatore	Descrizione	Formula
	aTick) 2. Se il prezzo di una posizione short (ordine di vendita sell) supera il prezzo stimato, MarginDiscount = Lots * ((PrezzoOrdine - PriceSettle) * PrezzoTick / GrandezzaTick) dove: ○ Pric	

Identificatore	Descrizione	Formula
	eSettle-prezzo stimato (clearing) della sessione precedente	

Identificatore	Descrizione	Formula
	dente; ○ PriceOrder(PrezzoOrdine)-posizioneprezzom	

Identificatore	Descrizione	Formula
	e d i o p p o n d e r a t o o p r e z z o d i a p e r t u r a f i s s a t o n e l l ' o r d i n e	

Identificatore	Descrizione	Formula
	richiesta); ○ TickPrice(PrezzoTick) - prezzotick(co	

Identificatore	Descrizione	Formula
	stodellavariazionedipreZzzodiunpuntoo) TickSize(Gr	

Identificatore	Descrizione	Formula
	andezzaTick) - grandezzatick(minimavariazionedist	

Identificatore	Descrizione	Formula
	e pdi pr ez zo)	
SYMBOL_CALC_MODE_EXCH_BONDS	Modalità Obbligazioni in borsa (exchange bonds) - calcolo del margine e del profitto per il trading di obbligazioni in borsa(stock exchange)	Margine: Lotti * GrandezzaContratto * ValoreDiFacciata * prezzo_open * / 100 Profitto: Lotti * prezzo_close * ValoreDiFacciata * Grandezza_Contratto + InteresseMaturato * Lotti * GrandezzaContratto
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	Modalità scambio	Margine: Lotti * GrandezzaContratto * PrezzoLast * Tasso_del_Margine

Identificatore	Descrizione	Formula
	MOEX in borsa - calcolo del margine e profitto per il trading di titoli (securities) borsa(stock) su MOEX	Profitto: (prezzo_close - prezzo_open) * Grandezza_Contratto * Lotti
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	Cambio modalità MOEX Bonds - calcolo del margine e del profitto per in trading obbligazioni(bonds) su	Margine: Lotti * GrandezzaContratto * ValoreDiFacciata * prezzo_open / 100 Profitto: Lotti * prezzo_close * ValoreDiFacciata * Grandezza_Contratto + InteresseMaturato * Lotti * GrandezzaContratto

Identificatore	Descrizione	Formula
	MOEX	
SYMBOL_CALC_MODE_SERV_COLLATERAL	Modalità collaterale - un simbolo viene utilizzato come attività non negoziabile su un conto di trading. Il valore di mercato di una posizione aperta è calcolato in base al volume, al prezzo corrente	Margine: no Profitto: no Valore del Margine: Lots * GrandezzaContratto * PrezzoDiMercato * LiqudityRate

Identificatore	Descrizione	Formula
	di mercato, alla dimensione del contratto e al rapporto di liquidità. Il valore è incluso in Assets, che vengono aggiunti ad Equity. Le posizioni aperte di tali simboli aumentano la quantità di Margine Libero e	

Identificatore	Descrizione	Formula
	vengono utilizzate come margine aggiuntivo (collaterale) per le posizioni aperte di strumenti negoziabili.	

Ci sono diversi modi di scambio di simboli. Le informazioni sulle modalità di trading di un certo simbolo si riflettono nei valori dell'enumerazione ENUM_SYMBOL_TRADE_MODE.

ENUM_SYMBOL_TRADE_MODE

Identificatore	Descrizione
SYMBOL_TRADE_MODE_DISABLED	Il trade è disabilitato per il simbolo
SYMBOL_TRADE_MODE_LONGONLY	Permesso solo per posizioni long
SYMBOL_TRADE_MODE_SHORTONLY	Permesso solo per posizioni short
SYMBOL_TRADE_MODE_CLOSEONLY	Permesso solo operazioni di chiusura posizioni
SYMBOL_TRADE_MODE_FULL	Non ci sono restrizioni trade

Possibili modalità di esecuzione di un affare per un certo simbolo sono definiti nell'enumerazione ENUM_SYMBOL_TRADE_EXECUTION.

ENUM_SYMBOL_TRADE_EXECUTION

Identificatore	Descrizione
SYMBOL_TRADE_EXECUTION_REQUEST	Esecuzione a richiesta
SYMBOL_TRADE_EXECUTION_INSTANT	Esecuzione istantanea
SYMBOL_TRADE_EXECUTION_MARKET	Esecuzione a mercato
SYMBOL_TRADE_EXECUTION_EXCHANGE	Esecuzione di scambio

Metodi di calcolo di swap al trasferimento posizione vengono specificati nell'enumerazione ENUM_SYMBOL_SWAP_MODE. Il metodo di calcolo di swap determina le unità di misura dei parametri SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT. Ad esempio, se gli swap vengono addebitati nella valuta di deposito client, allora i valori di tali parametri sono specificati come una somma di denaro nella valuta di deposito del cliente.

ENUM_SYMBOL_SWAP_MODE

Identificatore	Descrizione
SYMBOL_SWAP_MODE_DISABLED	Swap disabilitato (senza swap)
SYMBOL_SWAP_MODE_POINTS	Gli swap sono espressi in punti
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Gli swap vengono caricati in denaro nella valuta di base del simbolo
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Gli swap vengono caricati in denaro nella valuta margine del simbolo
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Gli swap vengono caricati in denaro nella valuta di deposito del cliente
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Gli swap vengono caricati come interesse specifico annuale del prezzo di uno strumento al calcolo dello swap (l'anno standard della banca è 360 giorni)
SYMBOL_SWAP_MODE_INTEREST_OPEN	Gli swap vengono caricati come interesse specifico annuale del prezzo di apertura della posizione (l'anno standard della banca è 360 giorni)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Gli swap vengono caricati dalla riapertura delle posizioni. Al termine di una giornata di trading la posizione viene chiusa. Il giorno dopo viene riaperto dai punti +/- specificati del prezzo di chiusura (parametri SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)
SYMBOL_SWAP_MODE_REOPEN_BID	Gli swap vengono caricati dalla riapertura delle posizioni. Al termine di una giornata di trading la posizione viene chiusa. Il giorno dopo viene

Identificatore	Descrizione
	riaperto dai punti +/- specificati del prezzo Bid corrente (parametri SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)

I valori dell'enumerazione ENUM_DAY_OF_WEEK vengono utilizzati per specificare i giorni della settimana.

ENUM_DAY_OF_WEEK

Identificatore	Descrizione
SUNDAY	Domenica
MONDAY	Lunedi
TUESDAY	Martedì
WEDNESDAY	Mercoledì
THURSDAY	Giovedi
FRIDAY	Venerdì
SATURDAY	Sabato

Un'opzione è un contratto che dà il diritto, ma non l'obbligo, di acquistare o vendere un bene sottostante (beni, titoli, futures, ecc.) ad un prezzo specificato prima o prima di una data specifica. Le seguenti enumerazioni descrivono le proprietà delle opzioni, tra cui il tipo di opzione e il diritto che ne derivano.

ENUM_SYMBOL_OPTION_RIGHT

Identificatore	Descrizione
SYMBOL_OPTION_RIGHT_CALL	Un'opzione di chiamata (call) ti consente di acquistare un asset(attività) ad un prezzo specificato
SYMBOL_OPTION_RIGHT_PUT	L'opzione di messa (put) ti consente di vendere un asset(attività) ad un prezzo specificato

ENUM_SYMBOL_OPTION_MODE

Identificatore	Descrizione
SYMBOL_OPTION_MODE_EUROPEAN	L'opzione europea può essere esercitata solo in una data specifica (scadenza, data di esecuzione, data di consegna)

Identificatore	Descrizione
SYMBOL_OPTION_MODE_AMERICAN	L'opzione americana può essere esercitata in qualsiasi giorno di trading prima o prima della scadenza. È specificato il periodo entro il quale un acquirente può esercitare l'opzione

Financial instruments are categorized by sectors of the economy. An economic sector is a part of economic activity which has specific characteristics, economic goals, functions and behavior, which allow separating this sector from other parts of the economy. ENUM_SYMBOL_SECTOR lists the economic sectors which a trading instruments can belong to.

ENUM_SYMBOL_SECTOR

ID	Description
SECTOR_UNDEFINED	Undefined
SECTOR_BASIC_MATERIALS	Basic materials
SECTOR_COMMUNICATION_SERVICES	Communication services
SECTOR_CONSUMER_CYCLICAL	Consumer cyclical
SECTOR_CONSUMER_DEFENSIVE	Consumer defensive
SECTOR_CURRENCY	Currencies
SECTOR_CURRENCY_CRYPTO	Cryptocurrencies
SECTOR_ENERGY	Energy
SECTOR_FINANCIAL	Finance
SECTOR_HEALTHCARE	Healthcare
SECTOR_INDUSTRIALS	Industrials
SECTOR_REAL_ESTATE	Real estate
SECTOR_TECHNOLOGY	Technology
SECTOR_UTILITIES	Utilities

Each financial instrument can be assigned to a specific type of industry or economy branch. An industry is a branch of an economy that produces a closely related set of raw materials, goods, or services. ENUM_SYMBOL_INDUSTRY lists industries which a trading instrument can belong to.

ENUM_SYMBOL_INDUSTRY

ID	Description
INDUSTRY_UNDEFINED	Undefined

ID	Description
Basic materials	
INDUSTRY_AGRICULTURAL_INPUTS	Agricultural inputs
INDUSTRY_ALUMINIUM	Aluminium
INDUSTRY_BUILDING_MATERIALS	Building materials
INDUSTRY_CHEMICALS	Chemicals
INDUSTRY_COKING_COAL	Coking coal
INDUSTRY_COPPER	Copper
INDUSTRY_GOLD	Gold
INDUSTRY_LUMBER_WOOD	Lumber and wood production
INDUSTRY_INDUSTRIAL_METALS	Other industrial metals and mining
INDUSTRY_PRECIOUS_METALS	Other precious metals and mining
INDUSTRY_PAPER	Paper and paper products
INDUSTRY_SILVER	Silver
INDUSTRY_SPECIALTY_CHEMICALS	Specialty chemicals
INDUSTRY_STEEL	Steel
Communication services	
INDUSTRY_ADVERTISING	Advertising agencies
INDUSTRY_BROADCASTING	Broadcasting
INDUSTRY_GAMING_MULTIMEDIA	Electronic gaming and multimedia
INDUSTRY_ENTERTAINMENT	Entertainment
INDUSTRY_INTERNET_CONTENT	Internet content and information
INDUSTRY_PUBLISHING	Publishing
INDUSTRY_TELECOM	Telecom services
Consumer cyclical	
INDUSTRY_APPAREL_MANUFACTURING	Apparel manufacturing
INDUSTRY_APPAREL_RETAIL	Apparel retail
INDUSTRY_AUTO_MANUFACTURERS	Auto manufacturers
INDUSTRY_AUTO_PARTS	Auto parts
INDUSTRY_AUTO_DEALERSHIP	Auto and truck dealerships
INDUSTRY_DEPARTMENT_STORES	Department stores

ID	Description
INDUSTRY_FOOTWEAR_ACCESSORIES	Footwear and accessories
INDUSTRY_FURNISHINGS	Furnishing, fixtures and appliances
INDUSTRY_GAMBLING	Gambling
INDUSTRY_HOME_IMPROV_RETAIL	Home improvement retail
INDUSTRY_INTERNET_RETAIL	Internet retail
INDUSTRY_LEISURE	Leisure
INDUSTRY_LODGING	Lodging
INDUSTRY_LUXURY_GOODS	Luxury goods
INDUSTRY_PACKAGING_CONTAINERS	Packaging and containers
INDUSTRY_PERSONAL_SERVICES	Personal services
INDUSTRY_RECREATIONAL_VEHICLES	Recreational vehicles
INDUSTRY_RESIDENT_CONSTRUCTION	Residential construction
INDUSTRY_RESORTS_CASINOS	Resorts and casinos
INDUSTRY_RESTAURANTS	Restaurants
INDUSTRY_SPECIALTY_RETAIL	Specialty retail
INDUSTRY_TEXTILE_MANUFACTURING	Textile manufacturing
INDUSTRY_TRAVEL_SERVICES	Travel services
Consumer defensive	
INDUSTRY_BEVERAGES_BREWERS	Beverages - Brewers
INDUSTRY_BEVERAGES_NON_ALCO	Beverages - Non-alcoholic
INDUSTRY_BEVERAGES_WINERIES	Beverages - Wineries and distilleries
INDUSTRY_CONFECTIONERS	Confectioners
INDUSTRY_DISCOUNT_STORES	Discount stores
INDUSTRY_EDUCATION_TRAINIG	Education and training services
INDUSTRY_FARM_PRODUCTS	Farm products
INDUSTRY_FOOD_DISTRIBUTION	Food distribution
INDUSTRY_GROCERY_STORES	Grocery stores
INDUSTRY_HOUSEHOLD_PRODUCTS	Household and personal products
INDUSTRY_PACKAGED_FOODS	Packaged foods
INDUSTRY_TOBACCO	Tobacco

ID	Description
Energy	
INDUSTRY_OIL_GAS_DRILLING	Oil and gas drilling
INDUSTRY_OIL_GAS_EP	Oil and gas extraction and processing
INDUSTRY_OIL_GAS_EQUIPMENT	Oil and gas equipment and services
INDUSTRY_OIL_GAS_INTEGRATED	Oil and gas integrated
INDUSTRY_OIL_GAS_MIDSTREAM	Oil and gas midstream
INDUSTRY_OIL_GAS_REFINING	Oil and gas refining and marketing
INDUSTRY_THERMAL_COAL	Thermal coal
INDUSTRY_URANIUM	Uranium
Finance	
INDUSTRY_EXCHANGE_TRADED_FUND	Exchange traded fund
INDUSTRY_ASSETS_MANAGEMENT	Assets management
INDUSTRY_BANKS_DIVERSIFIED	Banks - Diversified
INDUSTRY_BANKS_REGIONAL	Banks - Regional
INDUSTRY_CAPITAL_MARKETS	Capital markets
INDUSTRY_CLOSE_END_FUND_DEBT	Closed-End fund - Debt
INDUSTRY_CLOSE_END_FUND_EQUITY	Closed-end fund - Equity
INDUSTRY_CLOSE_END_FUND_FOREIGN	Closed-end fund - Foreign
INDUSTRY_CREDIT_SERVICES	Credit services
INDUSTRY_FINANCIAL_CONGLOMERATE	Financial conglomerates
INDUSTRY_FINANCIAL_DATA_EXCHANGE	Financial data and stock exchange
INDUSTRY_INSURANCE_BROKERS	Insurance brokers
INDUSTRY_INSURANCE_DIVERSIFIED	Insurance - Diversified
INDUSTRY_INSURANCE_LIFE	Insurance - Life
INDUSTRY_INSURANCE_PROPERTY	Insurance - Property and casualty
INDUSTRY_INSURANCE_REINSURANCE	Insurance - Reinsurance
INDUSTRY_INSURANCE_SPECIALTY	Insurance - Specialty
INDUSTRY_MORTGAGE_FINANCE	Mortgage finance
INDUSTRY_SHELL_COMPANIES	Shell companies
Healthcare	

ID	Description
INDUSTRY_BIOTECHNOLOGY	Biotechnology
INDUSTRY_DIAGNOSTICS_RESEARCH	Diagnostics and research
INDUSTRY_DRUGS_MANUFACTURERS	Drugs manufacturers - general
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	Drugs manufacturers - Specialty and generic
INDUSTRY_HEALTHCARE_PLANS	Healthcare plans
INDUSTRY_HEALTH_INFORMATION	Health information services
INDUSTRY_MEDICAL_FACILITIES	Medical care facilities
INDUSTRY_MEDICAL_DEVICES	Medical devices
INDUSTRY_MEDICAL_DISTRIBUTION	Medical distribution
INDUSTRY_MEDICAL_INSTRUMENTS	Medical instruments and supplies
INDUSTRY_PHARM_RETAILERS	Pharmaceutical retailers
Industrials	
INDUSTRY_AEROSPACE_DEFENSE	Aerospace and defense
INDUSTRY_AIRLINES	Airlines
INDUSTRY_AIRPORTS_SERVICES	Airports and air services
INDUSTRY_BUILDING_PRODUCTS	Building products and equipment
INDUSTRY_BUSINESS_EQUIPMENT	Business equipment and supplies
INDUSTRY_CONGLOMERATES	Conglomerates
INDUSTRY_CONSULTING_SERVICES	Consulting services
INDUSTRY_ELECTRICAL_EQUIPMENT	Electrical equipment and parts
INDUSTRY_ENGINEERING_CONSTRUCTION	Engineering and construction
INDUSTRY_FARM_HEAVY_MACHINERY	Farm and heavy construction machinery
INDUSTRY_INDUSTRIAL_DISTRIBUTION	Industrial distribution
INDUSTRY_INFRASTRUCTURE_OPERATIONS	Infrastructure operations
INDUSTRY_FREIGHT_LOGISTICS	Integrated freight and logistics
INDUSTRY_MARINE_SHIPPING	Marine shipping
INDUSTRY_METAL_FABRICATION	Metal fabrication
INDUSTRY_POLLUTION_CONTROL	Pollution and treatment controls
INDUSTRY_RAILROADS	Railroads
INDUSTRY_RENTAL_LEASING	Rental and leasing services

ID	Description
INDUSTRY_SECURITY_PROTECTION	Security and protection services
INDUSTRY_SPEALITY_BUSINESS_SERVICES	Specialty business services
INDUSTRY_SPEALITY_MACHINERY	Specialty industrial machinery
INDUSTRY_STUFFING_EMPLOYMENT	Stuffing and employment services
INDUSTRY_TOOLS_ACCESSORIES	Tools and accessories
INDUSTRY_TRUCKING	Trucking
INDUSTRY_WASTE_MANAGEMENT	Waste management
Real estate	
INDUSTRY_REAL_ESTATE_DEVELOPMENT	Real estate - Development
INDUSTRY_REAL_ESTATE_DIVERSIFIED	Real estate - Diversified
INDUSTRY_REAL_ESTATE_SERVICES	Real estate services
INDUSTRY_REIT_DIVERSIFIED	REIT - Diversified
INDUSTRY_REIT_HEALTCARE	REIT - Healthcase facilities
INDUSTRY_REIT_HOTEL_MOTEL	REIT - Hotel and motel
INDUSTRY_REIT_INDUSTRIAL	REIT - Industrial
INDUSTRY_REIT_MORTAGE	REIT - Mortgage
INDUSTRY_REIT_OFFICE	REIT - Office
INDUSTRY_REIT_RESIDENTAL	REIT - Residential
INDUSTRY_REIT_RETAIL	REIT - Retail
INDUSTRY_REIT_SPECIALITY	REIT - Specialty
Technology	
INDUSTRY_COMMUNICATION_EQUIPMENT	Communication equipment
INDUSTRY_COMPUTER_HARDWARE	Computer hardware
INDUSTRY_CONSUMER_ELECTRONICS	Consumer electronics
INDUSTRY_ELECTRONIC_COMPONENTS	Electronic components
INDUSTRY_ELECTRONIC_DISTRIBUTION	Electronics and computer distribution
INDUSTRY_IT_SERVICES	Information technology services
INDUSTRY_SCIENTIFIC_INSTRUMENTS	Scientific and technical instruments
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	Semiconductor equipment and materials
INDUSTRY_SEMICONDUCTORS	Semiconductors

ID	Description
INDUSTRY_SOFTWARE_APPLICATION	Software - Application
INDUSTRY_SOFTWARE_INFRASTRUCTURE	Software - Infrastructure
INDUSTRY_SOLAR	Solar
Utilities	
INDUSTRY_UTILITIES_DIVERSIFIED	Utilities - Diversified
INDUSTRY_UTILITIES_POWERPRODUCERS	Utilities - Independent power producers
INDUSTRY_UTILITIES_RENEWABLE	Utilities - Renewable
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	Utilities - Regulated electric
INDUSTRY_UTILITIES_REGULATED_GAS	Utilities - Regulated gas
INDUSTRY_UTILITIES_REGULATED_WATER	Utilities - Regulated water
INDUSTRY_UTILITIES_FIRST	Start of the utilities services types enumeration. Corresponds to INDUSTRY_UTILITIES_DIVERSIFIED.
INDUSTRY_UTILITIES_LAST	End of the utilities services types enumeration. Corresponds to INDUSTRY_UTILITIES_REGULATED_WATER.

Proprietà Account

Per ottenere informazioni sul corrente account ci sono diverse funzioni: AccountInfoInteger(), AccountInfoDouble() e AccountInfoString(). I valori dei parametri delle funzioni possono accettare i valori delle corrispondenti enumerazioni ENUM_ACCOUNT_INFO.

Per la funzione AccountInfoInteger()

ENUM_ACCOUNT_INFO_INTEGER

Identificatore	Descrizione	Tipo
ACCOUNT_LOGIN	Numero di account	long
ACCOUNT_TRADE_MODE	Modalità trade dell' Account	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	Leva dell'account	long
ACCOUNT_LIMIT_ORDERS	Numero massimo consentito di ordini pendenti attivi	int
ACCOUNT_MARGIN_SO_MODE	Modalità per l'impostazione del margine minimo consentito	ENUM_ACCOUNT_STOPOUT_MODE
ACCOUNT_TRADE_ALLOWED	Trade consentito per il corrente account	bool

Identificatore	Descrizione	Tipo
ACCOUNT_TRADE_EXPERT	Trade consentito per un Expert Advisor	bool
ACCOUNT_MARGIN_MODE	Modalità di calcolo del margine	ENUM_ACCOUNT_MARGIN_MODE
ACCOUNT_CURRENCY_DIGITS	Il numero di posizioni decimali in valuta dell'account che sono necessarie per una visualizzazione accurata dei risultati di trading	int
ACCOUNT_FIFO_CLOSE	Un'indicazione che mostra che le posizioni possono essere chiuse solo	bool

Identificatore	Descrizione	Tipo
	dalla regola FIFO. Se il valore della proprietà è impostato su true, allora le posizioni di ciascun simbolo verranno chiuse nello stesso ordine in cui vengono aperte, a partire da quella più vecchia. Nel caso di un tentativo di chiudere posizioni in un ordine diverso, il trader riceverà un errore	

Identificatore	Descrizione	Tipo
	appropriato. Per gli account con la modalità di contabilizzazione della posizione senza copertura (ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), il valore della proprietà è sempre false.	

Per la funzione AccountInfoDouble()

ENUM_ACCOUNT_INFO_DOUBLE

Identificatore	Descrizione	Tipo
ACCOUNT_BALANCE	Bilancio dell'account in valuta di deposito	double
ACCOUNT_CREDIT	Credito dell'account nella	double

Identificatore	Descrizione	Tipo
	valuta di deposito	
ACCOUNT_PROFIT	Profitto attuale dell'account in valuta di deposito	double
ACCOUNT_EQUITY	Equità nella valuta di deposito	double
ACCOUNT_MARGIN	Margine utilizzzato dell'account nella valuta di deposito	double
ACCOUNT_MARGIN_FREE	Margine libero dell'account nella valuta di deposito	double
ACCOUNT_MARGIN_LEVEL	Livello di margine dell'account in percentuale	double
ACCOUNT_MARGIN_SO_CALL	Livello di margin call. A seconda dell'impostazione ACCOUNT_MARGIN_SO_MODE è espressa in percentuale o in valuta di deposito	double

Identificatore	Descrizione	Tipo
ACCOUNT_MARGIN_SO_SO	Livello di stop out del margine. A seconda dell'impostazione ACCOUNT_MARGIN_SO_MODE è espressa in percentuale o in valuta di deposito	double
ACCOUNT_MARGIN_INITIAL	Initial margin. L'importo riservato su un account per coprire il margine di tutti gli ordini in sospeso	double
ACCOUNT_MARGIN_MAINTENANCE	Margine di mantenimento Il capitale minimo riservato su un account per coprire l'importo minimo di tutte le posizioni aperte	double
ACCOUNT_ASSETS	Le correnti attività(assets) di un account	double
ACCOUNT_LIABILITIES	Le correnti passività(li	double

Identificatore	Descrizione	Tipo
	abilities) di un account	
ACCOUNT_COMMISSION_BLOCKED	L'importo attuale della commissione bloccata di un account	double

Per la funzione AccountInfoString()

ENUM_ACCOUNT_INFO_STRING

Identificatore	Descrizione	Tipo
ACCOUNT_NAME	Nome del cliente	string
ACCOUNT_SERVER	Nome del trade server	string
ACCOUNT_CURRENCY	Valuta dell'account	string
ACCOUNT_COMPANY	Nome della società che fornisce l'account	string

Ci sono diversi tipi di account che possono essere aperti su un trade server. Il tipo di account in cui un programma mql5 è in esecuzione può essere scoperto mediante l'enumerazione ENUM_ACCOUNT_TRADE_MODE.

ENUM_ACCOUNT_TRADE_MODE

Identificatore	Descrizione
ACCOUNT_TRADE_MODE_DEMO	Account demo
ACCOUNT_TRADE_MODE_CONTEST	Account contest
ACCOUNT_TRADE_MODE_REAL	Account reale

Nel caso in cui l'equità non è sufficiente a mantenere le posizioni aperte, avviene la situazione di Stop Out, vale a dire che si verifica la chiusura forzata. Il livello di margine minimo al quale si verifica lo

Stop Out può essere impostato in percentuale o in termini monetari. Per individuare la modalità impostata per l'account utilizzare l'enumerazione ENUM_ACCOUNT_STOPOUT_MODE.

ENUM_ACCOUNT_STOPOUT_MODE

Identificatore	Descrizione
ACCOUNT_STOPOUT_MODE_PERCENT	Modalità di stop out dell' account in termini percentuali
ACCOUNT_STOPOUT_MODE_MONEY	Modalità di stop out dell'account in termini monetari

ENUM_ACCOUNT_MARGIN_MODE

Identifier	Description
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	Utilizzato per i mercati OTC per interpretare le posizioni nella modalità "netting" (può esistere una sola posizione per un simbolo). Il margine viene calcolato in base al tipo di simbolo (SYMBOL_TRADE_CALC_MODE).
ACCOUNT_MARGIN_MODE_EXCHANGE	Utilizzato per i mercati valutari. Il margine viene calcolato in base agli sconti specificati nelle impostazioni dei simboli. Gli sconti sono stabiliti dal broker, ma non meno dei valori impostati dall'exchange.
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	Utilizzato per i mercati valutari in cui sono possibili posizioni individuali (hedging, possono esistere più posizioni per un simbolo). Il margine viene calcolato in base al tipo di simbolo (SYMBOL_TRADE_CALC_MODE) tenendo conto del margine coperto (SYMBOL_MARGIN_HEDGED).

Un esempio di script che emette in output brevi informazioni sull'account.

```
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- Nome della società
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- Nome del cliente
   string name=AccountInfoString(ACCOUNT_NAME);
//--- Numero dell'account
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- Nome del server
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- Valuta dell' account
```

```
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- Account demo, contest o reale
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- Ora trasformiamo il valore dell'enumerazione in una forma comprensibile
   string trade_mode;
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
         trade_mode="contest";
         break;
      default:
         trade_mode="real";
         break;
     }
//--- Lo Stop Out è impostato in percentuale o in denaro
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- Ottiene il valore dei livelli quando si verificano il Margin Call e lo Stop Out
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- Mostra brevi informazioni sull'account
   PrintFormat("L'account del cliente '%s' #%d %s aperto '%s' sul server '%s'",
               nome,login, trade_mode,società,server);
   PrintFormat("Valuta dell'account - %s, i livelli di MarginCall e StopOut sono impos
                currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"percentage":" r
   PrintFormat("MarginCall=%G, StopOut=%G",margin_call,stop_out);
  }
```

Statistiche Testing

Dopo che il test è finito, vengono calcolati diversi parametri delle statistiche di risultati di trading. I valori dei parametri possono essere ottenuti utilizzando la funzione TesterStatistics(), specificando il parametro ID dall'enumerazione ENUM_STATISTICS.

Anche se due tipi di parametri (int e double) vengono utilizzati per il calcolo delle statistiche, la funzione restituisce tutti i valori in forma double. Tutti i valori statistici di tipo double vengono espressi nella valuta di deposito di default, se non diversamente specificato.

ENUM_STATISTICS

ID	Descrizione di un parametro statistico	Tipo
STAT_INITIAL_DEPOSIT	Il valore del deposito iniziale	double
STAT_WITHDRAWAL	Il denaro prelevato da un conto	double
STAT_PROFIT	Il profitto netto dopo il testing, la somma di STAT_GROSS_PROFIT e STAT_GROSS_LOSS (STAT_GROSS_LOSS è sempre inferiore o uguale a zero)	double
STAT_GROSS_PROFIT	Profitto totale netto, la somma di tutti i trades profittevoli (positivi). Il valore è maggiore o uguale a zero	double
STAT_GROSS_LOSS	Perdita totale, la somma di tutti i trades negativi. Il valore è minore o uguale a zero	double
STAT_MAX_PROFITTRADE	Massimo profitto - il valore più grande di tutti i trades profittevoli. Il valore è maggiore o uguale a zero	double
STAT_MAX_LOSSTRADE	Massima perdita - il valore più basso di tutti i trades perdenti. Il valore è minore o uguale a zero	double

ID	Descrizione di un parametro statistico	Tipo
STAT_CONPROFITMAX	Massimo profitto in una serie di trades profittevoli. Il valore è maggiore o uguale a zero	double
STAT_CONPROFITMAX_TRADES	Il numero di trades che viene formato STAT_CONPROFITMAX (massimo profitto in una serie di trades profittevoli)	int
STAT_MAX_CONWINS	Il profitto totale della più lunga serie di trades profittevoli	double
STAT_MAX_CONPROFIT_TRADES	Il numero di trades nella più lunga serie di trades profittevoli STAT_MAX_CONWINS	int
STAT_CONLOSSMAX	La perdita massima in una serie di trades perdenti. Il valore è minore o uguale a zero	double
STAT_CONLOSSMAX_TRADES	Il numero di transazioni che si sono formate STAT_CONLOSSMAX (perdita massima in una serie di trades perdenti)	int
STAT_MAX_CONLOSSES	La perdita totale della più lunga serie di trades perdenti	double
STAT_MAX_CONLOSS_TRADES	Il numero di trades nella più lunga serie di trades perdenti STAT_MAX_CONLOSSES	int
STAT_BALANCEMIN	Valore minimo del bilancio	double
STAT_BALANCE_DD	Drawdown massimo del bilancio in termini monetari. Nel processo di trading, un bilancio	double

ID	Descrizione di un parametro statistico	Tipo
	può avere numerosi drowdowns; qui viene preso il valore più grande	
STAT_BALANCEDD_PERCENT	Drawdown bilancio come percentuale che è stata registrata al momento del drawdown massimo del bilancio in termini monetari (STAT_BALANCE_DD).	double
STAT_BALANCE_DDREL_PERCENT	Drawdown bilancio massimo in percentuale. Nel processo di trading, un bilancio può avere numerosi drowdowns, per ciascuno dei quali viene calcolato il relativo valore drowdown in percentuale. Viene restituito il maggior valore	double
STAT_BALANCE_DD_RELATIVE	Drawdown bilancio in termini monetari che è stato registrato al momento del drowdown massimo del bilancio in percentuale (STAT_BALANCE_DDREL_PERCENT).	double
STAT_EQUITYMIN	Valore minimo dell'equità	double
STAT_EQUITY_DD	Valore massimo dell' rquità in termini monetari. Nel processo di trading, numerosi drowdowns possono essere visualizzati sul sull'equità; qui viene preso il valore più grande.	double

ID	Descrizione di un parametro statistico	Tipo
STAT_EQUITYDD_PERCENT	Drawdown in percentuale che è stato registrato al momento del drowdown massimo dell' equità in termini monetari (STAT_EQUITY_DD).	double
STAT_EQUITY_DDREL_PERCENT	Drawdown massomo dell'equità come percentuale. Nel processo di trading, un'equità può avere numerosi drawdowns, per ciascuno dei quali viene calcolato il valore drowdown relativo in percentuale. Viene restituito il maggior valore	double
STAT_EQUITY_DD_RELATIVE	Drowdown dell'equità in termini monetari che sono stati registrati al momento del drowdown massimo dell'equità, in percentuale(STAT_EQUITY_DDREL_PERCENT).	double
STAT_EXPECTED_PAYOFF	Payoff atteso	double
STAT_PROFIT_FACTOR	Fattore profitto, uguale al rapporto di STAT_GROSS_PROFIT / STAT_GROSS_LOSS. Se STAT_GROSS_LOSS=0, il fattore di profitto è pari a DBL_MAX	double
STAT_RECOVERY_FACTOR	Fattore di recupero, pari al rapporto di STAT_PROFIT / STAT_BALANCE_DD	double
STAT_SHARPE_RATIO	Indice di Sharpe	double

ID	Descrizione di un parametro statistico	Tipo
STAT_MIN_MARGINLEVEL	Valore minimo del livello di margine	double
STAT_CUSTOM_ONTESTER	Il valore del criterio di ottimizzazione personalizzata calcolato, restituito dalla funzione OnTester()	double
STAT_DEALS	Il numero di affari	int
STAT_TRADES	Il numero di trades	int
STAT_PROFIT_TRADES	Trades profittevoli	int
STAT_LOSS_TRADES	Trades perdenti	int
STAT_SHORT_TRADES	Trades short	int
STAT_LONG_TRADES	Trades long	int
STAT_PROFIT_SHORTTRADES	Trades short profittevoli	int
STAT_PROFIT_LONGTRADES	Trades long profittevoli	int
STAT_PROFITTRADES_AVGCON	Lunghezza media di una serie di trades profittevoli	int
STAT_LOSSTRADES_AVGCON	Lunghezza media di una serie di trades perdenti	int
STAT_COMPLEX_CRITERION	Criterio di ottimizzazione complesso	

Costanti di Trade

Varie costanti utilizzate la programmazione di strategie di trading sono suddivise nei seguenti gruppi:

- Proprietà Database Storico - Riceve informazioni generali su un simbolo;
- Proprietà dell' ordine - Ottiene informazioni sugli ordini di trade;
- Proprietà della Posizione - Ottiene informazioni sulle posizioni attuali;
- Proprietà dell' Affare - Ottenere informazioni sull' Affare;
- Tipi di operazioni di trade - Descrizione delle operazioni di trade disponibili;
- Tipi di transazioni di Trade - Descrizione dei tipi di possibili di transazioni di trade;
- Ordini di Trade nel DOM – separazione degli ordini secondo la direzione dell'operazione richiesta

Proprietà del database della CroniStoria

Quando si accede alle timeseries viene usata la funzione SeriesInfoInteger() per ottenere ulteriori simbolo di informazione. L'identificatore di una proprietà obbligatoria viene passato come parametro della funzione. L'identificatore può essere uno dei valori di ENUM_SERIES_INFO_INTEGER.

ENUM_SERIES_INFO_INTEGER

Identificatore	Descrizione	Tipo
SERIES_BARS_COUNT	Conteggio barre per il simbolo-periodo per il momento attuale	long
SERIES_FIRSTDATE	La prima vera data per il simbolo-periodo per il momento attuale	datetime
SERIES_LASTBAR_DATE	Tempo di apertura dell'ultima barra del simbolo-periodo	datetime
SERIES_SERVER_FIRSTDATE	La prima data nello storico del simbolo sul server, indipendentemente dal periodo di tempo	datetime
SERIES_TERMINAL_FIRSTDATE	La prima data nello storico del simbolo nel terminale client, indipendentemente dal timeframe	datetime
SERIES_SYNCHRONIZED	Flag della data di sincronizzazione per il simbolo/periodo per il momento attuale	bool

Proprietà Ordini

Le richieste per eseguire le operazioni di trade vengono formalizzate come ordini. Ogni ordine ha una serie di proprietà per la lettura. Le informazioni su di essi possono essere ottenute utilizzando le funzioni OrderGet...() e HistoryOrderGet...().

Per le funzioni OrderGetInteger() ed HistoryOrderGetInteger()

ENUM_ORDER_PROPERTY_INTEGER

Identificatore	Descrizione	Tipo
ORDER_TICKET	Ticket Ordine. Numero unico assegnato ad ogni ordine	long
ORDER_TIME_SETUP	Setup dell'orario dell' Ordine	datetime
ORDER_TYPE	Tipo di ordine	ENUM_ORDER_TYPE
ORDER_STATE	Stato dell'ordine	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	Orario di espirazione dell'ordine	datetime
ORDER_TIME_DONE	Orario di esecuzione o eliminazione dell' Ordine	datetime
ORDER_TIME_SETUP_MSC	L'orario di piazzamento dell'esecuzione di un ordine in millisecondi dal 01.01.1970	long
ORDER_TIME_DONE_MSC	Orario di esecuzione/eliminazione di ordini in millisecondi dal 01.01.1970	long
ORDER_TYPE_FILLING	Tipo di riempimento dell'ordine	ENUM_ORDER_TYPE_FILLING

Identificatore	Descrizione	Tipo
ORDER_TYPE_TIME	Durata dell'ordine	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	ID di un Expert Advisor che ha piazzato l'ordine (progettato per garantire che ogni Expert Advisor collochi il proprio numero unico)	long
ORDER_REASON	La ragione o la fonte per l'invio di un ordine	ENUM_ORDER_REASON
ORDER_POSITION_ID	Identificatore Posizione che è impostato in un ordine non appena viene eseguito. Ogni ordine eseguito risulta in un affare che apre o modifica una posizione già esistente. L'identificatore di esattamente questa posizione viene impostato ad ordine eseguito, in questo momento.	long
ORDER_POSITION_BY_ID	Identificatore di una posizione opposta	long

Identificatore	Descrizione	Tipo
	utilizzata per la chiusura da ORDER_TYPE_CLOSE_BY	

Per le funzioni OrderGetDouble() e HistoryOrderGetDouble()

ENUM_ORDER_PROPERTY_DOUBLE

Identificatore	Descrizione	Tipo
ORDER_VOLUME_INITIAL	Volume iniziale dell'ordine	double
ORDER_VOLUME_CURRENT	Volume corrente dell'ordine	double
ORDER_PRICE_OPEN	Prezzo specificato nell'ordine	double
ORDER_SL	Valore Stop Loss	double
ORDER_TP	Valore Take Profit	double
ORDER_PRICE_CURRENT	Il prezzo attuale del simbolo dell'ordine	double
ORDER_PRICE_STOPLIMIT	Il prezzo Limit dell' ordine per l'ordine StopLimit	double

Per le funzioni OrderGetString() e HistoryOrderGetString()

ENUM_ORDER_PROPERTY_STRING

Identificatore	Descrizione	Tipo
ORDER_SYMBOL	Smbolo dell'ordine	string
ORDER_COMMENT	Commento all'ordine	string

Quando si invia una richiesta di trade usando la funzione OrderSend(), alcune operazioni richiedono l'indicazione del tipo di ordine. Il tipo di ordine viene specificato nel campo *tipo* della struttura speciale MqlTradeRequest, e può accettare valori dell'enumerazione ENUM_ORDER_TYPE.

ENUM_ORDER_TYPE

Identificatore	Descrizione
ORDER_TYPE_BUY	Ordine di mercato Buy
ORDER_TYPE_SELL	Ordine di mercato Sell
ORDER_TYPE_BUY_LIMIT	Ordine pendente Buy Limit
ORDER_TYPE_SELL_LIMIT	Ordine pendente Sell Limit
ORDER_TYPE_BUY_STOP	Ordine pendente Buy Stop
ORDER_TYPE_SELL_STOP	Ordine pendente Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Dopo aver raggiunto il prezzo dell' ordine, un ordine pentende Buy Limit viene piazzato al prezzo StopLimit
ORDER_TYPE_SELL_STOP_LIMIT	Dopo aver raggiunto il prezzo dell'ordine, un ordine pendente Sell Limit viene piazzato al prezzo StopLimit
ORDER_TYPE_CLOSE_BY	Order to close a position by an opposite one

Ogni ordine ha uno status che descrive il suo stato. Per ottenere informazioni, utilizzare OrderGetInteger() oppure HistoryOrderGetInteger() con il modificatore ORDER_STATE. I valori consentiti vengono memorizzati nell' enumerazione ENUM_ORDER_STATE.

ENUM_ORDER_STATE

Identificatore	Descrizione
ORDER_STATE_STARTED	Ordine controllato, ma non ancora accettato dal broker
ORDER_STATE_PLACED	Ordine accettato
ORDER_STATE_CANCELED	Ordine annullato dal client
ORDER_STATE_PARTIAL	Ordine parzialmente eseguito
ORDER_STATE_FILLED	Ordine pienamente eseguito
ORDER_STATE_REJECTED	Ordine rigettato
ORDER_STATE_EXPIRED	Ordine espirato
ORDER_STATE_REQUEST_ADD	L' ordine è stato registrato (piazzato al trading system)
ORDER_STATE_REQUEST_MODIFY	L'ordine è stato modificato (cambio dei suoi parametri)
ORDER_STATE_REQUEST_CANCEL	L'ordine è stato eliminato (eliminato dal trading system)

Quando si invia una richiesta di trade per l'esecuzione **in tempo reale** (ora in vigore), occorre specificare il prezzo e il volume di acquisto/vendita richiesto. Inoltre, tieni presente che i mercati finanziari non forniscono alcuna garanzia che l'intero volume richiesto sia disponibile per un determinato strumento finanziario al prezzo desiderato. Pertanto, le operazioni di trading in tempo reale sono regolate utilizzando le modalità di esecuzione di prezzo e volume. Le modalità, o criteri di esecuzione, definiscono le regole per i casi in cui il prezzo è cambiato o il volume richiesto non può essere completamente soddisfatto **in quel momento**.

Le modalità di esecuzione dei prezzi possono essere ottenute dalla proprietà del simbolo SYMBOL_TRADE_EXEMODE contenente la combinazione di flag dall'enumerazione ENUM_SYMBOL_TRADE_EXECUTION.

Modalità di esecuzione	Descrizione	Valore in ENUM_SYMBOL_TRADE_EXECUTION
Modalità di esecuzione (Request Execution)	Esecuzione di un ordine di mercato al prezzo precedentemente ricevuto dal broker. I prezzi per un certo ordine di mercato sono richiesti dal broker prima che l'ordine venga inviato. Dopo aver ricevuto i prezzi, l'esecuzione	SYMBOL_TRADE_EXECUTION_REQUEST

Modalità di esecuzione	Descrizione	Valore in ENUM_SYMBOL_TRADE_EXECUTION
	dell'ordine al prezzo indicato può essere confermato o respinto.	
Esecuzione Immediata (Instant Execution)	Esecuzione immediata di un ordine di mercato al prezzo specificato. Quando si invia una richiesta di trade da eseguire, la piattaforma aggiunge automaticamente i prezzi correnti all'ordine. • Se i	SYMBOL_TRADE_EXECUTION_INSTANT

Modalità di esecuzione	Descrizione	Valore in ENUM_SYMBOL_TRADE_EXECUTION
	• Il broker accetta il prezzo, l'ordine viene eseguito. • Se il	

Modalità di esecuzione	Descrizione	Valore in ENUM_SYMBOL_TRADE_EXECUTION
	broker non accetta il prezzo richiesto, viene inviato un	

Modalità di esecuzione	Descrizione	Valore in ENUM_SYMBOL_TRADE_EXECUTION
	"Requote" – il broker restituisce i prezzi, alla quale	

Modalità di esecuzione	Descrizione	Valore in ENUM_SYMBOL_TRADE_EXECUTION
	estoordinepuòessereeseguito.	
Esecuzione a Mercato (Market Execution)	Un broker prende una decisione circa il prezzo di esecuzione dell'ordine senza alcuna discussione aggiuntiva con	SYMBOL_TRADE_EXECUTION_MARKET

Modalità di esecuzione	Descrizione	Valore in ENUM_SYMBOL_TRADE_EXECUTION
	il trader. L'invio dell'ordine in tale modalità significa il consenso anticipato alla sua esecuzione a questo prezzo.	
Esecuzione in Borsa (Exchange Execution)	Le operazioni di trade sono eseguite ai prezzi delle attuali offerte di mercato.	SYMBOL_TRADE_EXECUTION_EXCHANGE

La politica di inserimento del volume viene specificata nella proprietà dell'ordine ORDER_TYPE_FILLING e può contenere solo i valori dell'enumerazione ENUM_ORDER_TYPE_FILLING

Politica di inserimento	Descrizione	Valore in ENUM_ORDER_TYPE_FILLING
Tutto o Niente	Un ordine può essere eseguito solo nel	ORDER_FILLING_FOK

Politica di inserimento	Descrizione	Valore in ENUM_ORDER_TYPE_FILLING
	volume specificato. Se l'ammontare necessario di uno strumento finanziario non è attualmente disponibile sul mercato, l'ordine non verrà eseguito. Il volume desiderato può essere costituito da diverse offerte disponibili. La possibilità di utilizzare ordini FOK è	

Politica di inserimento	Descrizione	Valore in ENUM_ORDER_TYPE_FILLING
	determinato dal server di trading.	
Immediato o Annulla	Un trader accetta di eseguire un contratto con il massimo volume disponibile sul mercato all'interno di quello indicato nell'ordine. Se la richiesta non può essere inserita completamente, verrà eseguito un ordine con il volume disponi	ORDER_FILLING_IOC

Politica di inserimento	Descrizione	Valore in ENUM_ORDER_TYPE_FILLING
	bile e il volume rimanente verrà cancellato. La possibilità di utilizzare ordini IOC è determinato dal server di trading.	
Passivo (Book or Cancel)	L'ordine BoC presuppone che l'ordine possa essere effettuato solo nel Depth of Market e non possa essere eseguito immediatamente. Se l'ordine	ORDER_FILLING_BOC

Politica di inserimento	Descrizione	Valore in ENUM_ORDER_TYPE_FILLING
	può essere eseguito immediatamente quando piazzato, allora viene annullato. Infatti, la politica BoC garantisce che il prezzo dell'ordine effettuato sarà peggiore dell'attuale mercato. Gli ordini BoC sono utilizzati per implementare la negoziazione passiva, in modo che	

Politica di inserimento	Descrizione	Valore in ENUM_ORDER_TYPE_FILLING
	l'ordine non venga eseguito immediatamente al momento dell'immissione e non influisca sulla liquidità corrente. Sono supportati solo gli ordini limit e stop limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT, ORDER_TYPE_SELL_STOP_	

Politica di inserimento	Descrizione	Valore in ENUM_ORDER_TYPE_FILLING
	LIMIT).	
Ritorno	In caso di inserimento parziale, un ordine con volume residuo non viene annullato ma ulteriormente elaborato. Gli ordini con ritorno non sono consentiti nella modalità Esecuzione a Mercato (market execution — SYMBOL_TRADE_EXECUTION_MARKET).	ORDER_FILLING_RETURN

Quando si invia una richiesta di trade utilizzando la funzione OrderSend() , la politica di esecuzione del volume necessario può essere impostato nel campo *type_filling*, ovvero nella speciale struttura MqlTradeRequest. Sono disponibili i valori dell'enumerazione ENUM_ORDER_TYPE_FILLING. Per ottenere il valore della proprietà in uno specifico ordine attivo/completato, utilizzare le funzioni OrderGetInteger() o HistoryOrderGetInteger() con il modificatore ORDER_TYPE_FILLING.

Prima di inviare un ordine con esecuzione in tempo reale, per la corretta impostazione del valore (tipo di esecuzione del volume) ORDER_TYPE_FILLING, è possibile utilizzare la funzione SymbolInfoInteger() con ogni strumento finanziario per ottenere il valore della proprietà SYMBOL_FILLING_MODE, che mostra i tipi di esecuzione del volume consentiti per il simbolo come combinazione di flag. L'inserimento del tipo ORDER_FILLING_RETURN è sempre abilitato ad eccezione della modalità "Esecuzione a Mercato" (SYMBOL_TRADE_EXECUTION_MARKET).

L'uso dei tipi di inserimento a seconda della modalità di esecuzione può essere mostrato come nella seguente tabella:

Tipo di Esecuzione\Politica d'Inserimento	Tutto o Niente (FOK ORDER_FILLING_FOK)	Immediato o Annulla (IOC ORDER_FILLING_IOC)	Ritorno (Return ORDER_FILLING_RETURN)
Esecuzione Immediata (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (indipendentemente dall'impostazione di un simbolo)	+ (indipendentemente dall'impostazione di un simbolo)	+ (sempre)
Richiesta di Esecuzione SYMBOL_TRADE_EXECUTION_REQUEST	+ (indipendentemente dall'impostazione di un simbolo)	+ (indipendentemente dall'impostazione di un simbolo)	+ (sempre)
Esecuzione a Mercato SYMBOL_TRADE_EXECUTION_MARKET	+ (impostato nelle impostazioni del simbolo)	+ (impostato nelle impostazioni del simbolo)	- (disabilitato indipendentemente dalle impostazioni del simbolo)
Esecuzione in Borsa SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (impostato nelle impostazioni del simbolo)	+ (impostato nelle impostazioni del simbolo)	+ (sempre)

In caso di ordini pendenti, il tipo di inserimento ORDER_FILLING_RETURN deve essere utilizzato indipendentemente dal tipo di esecuzione (SYMBOL_TRADE_EXEMODE), poiché tali ordini non sono destinati per l'esecuzione al momento dell'invio. Quando si utilizzano gli ordini pendenti, un trader accetta in anticipo che, quando le condizioni per un contratto su questo ordine sono soddisfatte, il broker utilizzerà il tipo di inserimento supportato dallo scambio.

Il periodo di validità dell'ordine può essere impostato nel campo *type_time* della speciale struttura MqlTradeRequest durante l'invio di una richiesta di trade con la funzione OrderSend(). I valori dell'enumerazione ENUM_ORDER_TYPE_TIME sono ammessi. Per ottenere il valore di questa proprietà utilizzare la funzione OrderGetInteger() oppure HistoryOrderGetInteger() con il modificatore ORDER_TYPE_TIME.

ENUM_ORDER_TYPE_TIME

Identificatore	Descrizione
ORDER_TIME_GTC	Buona fino a cancellazione ordine
ORDER_TIME_DAY	Buono sino al corrente ordine d giorno di trade
ORDER_TIME_SPECIFIED	Buona fino ad espirazione ordine
ORDER_TIME_SPECIFIED_DAY	L'ordine sarà efficace fino a 23:59:59 del giorno specificato. Se questo orario è al di fuori di una sessione di trading, l'ordine scade nel più vicino orario di trading.

La ragione per l'immissione dell'ordine è contenuta nella proprietà ORDER_REASON. Un ordine può essere posizionato da un programma MQL5, da un'applicazione mobile, a seguito di StopOut, ecc. I valori possibili di ORDER_REASON sono descritti nell'enumerazione ENUM_ORDER_REASON.

ENUM_ORDER_REASON

Identificatore	Descrizione
ORDER_REASON_CLIENT	L'ordine è stato piazzato da un terminale desktop
ORDER_REASON_MOBILE	L'ordine è stato piazzato da un'applicazione mobile
ORDER_REASON_WEB	L'ordine è stato piazzato da una piattaforma web
ORDER_REASON_EXPERT	L'ordine è stato piazzato da un programma MQL5, cioè da un Expert Advisor o da uno script
ORDER_REASON_SL	L'ordine è stato piazzato in seguito all'attivazione dello Stop Loss
ORDER_REASON_TP	L'ordine è stato piazzato in seguito all'attivazione del Take Profit
ORDER_REASON_SO	L'ordine è stato piazzato in seguito all'attivazione dell'evento Stop Out

Proprietà Posizione

L' esecuzione di operazioni di trade provoca l'apertura di una posizione, cambiando il suo volume e/o direzione, o la sua scomparsa. Le operazioni di trade vengono effettuate sulla base di ordini, inviati dalla funzione OrderSend() sottoforma di richieste di trade. Per ogni strumento finanziario (simbolo) è possibile solo una posizione aperta. Una posizione ha un insieme di proprietà disponibili per la lettura da parte delle funzioni PositionGet...().

Per la funzione PositionGetInteger()

ENUM_POSITION_PROPERTY_INTEGER

Identificatore	Descrizione	Tipo
POSITION_TICKET	Ticket Posizione. Numero unico assegnato a ciascuna posizione di nuova apertura. Di solito corrisponde al ticket di un ordine utilizzato per aprire la posizione, tranne quando il ticket viene modificato a seguito di operazioni di servizio sul server, ad esempio quando si carica lo swap con riapertura posizione. Per trovare un ordine utilizzato per aprire una posizione, applicare la proprietà POSITION_IDENTIFIER. POSITION_TICKET il valore corrisponde a MqlTradeRequest::position.	long

Identificatore	Descrizione	Tipo
POSITION_TIME	Orario di apertura della posizione	datetime
POSITION_TIME_MSC	Orario di apertura della posizione in millisecondi dal 01.01.1970	long
POSITION_TIME_UPDATE	Orario di cambio della posizione	datetime
POSITION_TIME_UPDATE_MSC	Orario di cambio della posizione in millisecondi dal 01.01.1970	long
POSITION_TYPE	Tipo di posizione	ENUM_POSITION_TYPE
POSITION_MAGIC	Numero magico della posizione (vedi ORDER_MAGIC)	long
POSITION_IDENTIFIER	L'identificatore di posizione è un numero unico che viene assegnato ad ogni posizione nuovamente aperta e non cambia durante l'intero periodo di vita della posizione. Il turnover della posizione non cambia il suo identificatore. L'identificatore posizione è specificato in ogni ordine (ORDER_POSITION_ID) ed affare (DEAL_POSITION_ID) usati per aprirla, modificarla o chiuderla. Usare questa proprietà per cercare ordini	long

Identificatore	Descrizione	Tipo
	ed affari relativi alla posizione. Quando si inverte una posizione in modalità netting (usando un singolo trade in/out), POSITION_IDENTIFIER non cambia. Comunque, POSITION_TICKET viene rimpiazzato con il ticket dell'ordine che porta all'inversione. L'inversione della posizione non viene concessa in modalità hedging.	
POSITION_REASON	La ragione per aprire una posizione	ENUM_POSITION_REASON

Per la funzione PositionGetDouble()

ENUM_POSITION_PROPERTY_DOUBLE

Identificatore	Descrizione	Tipo
POSITION_VOLUME	Volume della posizione	double
POSITION_PRICE_OPEN	Prezzo di apertura della posizione	double
POSITION_SL	Livello di Stop Loss della posizione aperta	double
POSITION_TP	Livello di Take Profit della posizione aperta	double
POSITION_PRICE_CURRENT	Prezzo corrente del simbolo della posizione	double

Identificatore	Descrizione	Tipo
POSITION_SWAP	Swap cumulativo	double
POSITION_PROFIT	Profitto corrente	double

Per la funzione PositionGetString()

ENUM_POSITION_PROPERTY_STRING

Identificatore	Descrizione	Tipo
POSITION_SYMBOL	Simbolo della posizione	string
POSITION_COMMENT	Commento della Posizione	string
POSITION_EXTERNAL_ID	Identificatore posizione in un sistema di trading esterno (sull' Exchange)	string

La direzione di una posizione aperta (buy o sell) è definita dal valore dell'enumerazione ENUM_POSITION_TYPE. Per ottenere il tipo di una posizione aperta utilizzare la funzione PositionGetInteger() con il modificatore POSITION_TYPE.

ENUM_POSITION_TYPE

Identificatore	Descrizione
POSITION_TYPE_BUY	Buy
POSITION_TYPE_SELL	Sell

La ragione per aprire una posizione è contenuta nella proprietà POSITION_REASON. Una posizione può essere a seguito dell'attivazione di un ordine da un terminale desktop, da un'applicazione mobile, da un Expert Advisor, ecc. I possibili valori di POSITION_REASON sono descritti nell'enumerazione ENUM_POSITION_REASON.

ENUM_POSITION_REASON

Identificatore	Descrizione
POSITION_REASON_CLIENT	La posizione è stata aperta a seguito dell'attivazione di un ordine piazzato da un terminale desktop
POSITION_REASON_MOBILE	La posizione è stata aperta in seguito all'attivazione di un ordine piazzato da un'applicazione mobile

Identificatore	Descrizione
POSITION_REASON_WEB	La posizione è stata aperta a seguito dell'attivazione di un ordine piazzato dalla piattaforma web
POSITION_REASON_EXPERT	La posizione è stata aperta a seguito dell'attivazione di un ordine piazzato da un programma MQL5, ad esempio un Expert Advisor o uno Script

Proprietà Affari(Deal)

Un affare è il riflesso del fatto dell'esecuzione di un' operazione di trade basata su un ordine che contiene una richiesta di trade. Each trade is described by properties that allow to obtain information about it. Per leggere i valori delle proprietà, vengono utilizzate le funzioni del tipo HistoryDealGet...(), che restituiscono valori delle enumerazioni corrispondenti.

Per la funzione HistoryDealGetInteger()

ENUM_DEAL_PROPERTY_INTEGER

Identificatore	Descrizione	Tipo
DEAL_TICKET	Ticket Affare. Numero unico assegnato ad ogni affare	long
DEAL_ORDER	Affare numero d'ordine	long
DEAL_TIME	Orario affare	datetime
DEAL_TIME_MSC	Il tempo di esecuzione di un affare in millisecondi dal 01.01.1970	long
DEAL_TYPE	Tipo Affare	ENUM_DEAL_TYPE
DEAL_ENTRY	Entry dell'affare - ingresso in, ingresso out, inversione	ENUM_DEAL_ENTRY
DEAL_MAGIC	Numero magico dell' affare (vedi ORDER_MAGIC)	long
DEAL_REASON	La ragione o la fonte per l'esecuzione degli affari	ENUM_DEAL_REASON
DEAL_POSITION_ID	Identificatore di posizione, nell' apertura, modifica o chiusura di ciò a cui questo affare prende parte. Ogni posizione ha un identificatore univoco che viene assegnato a tutti gli affari eseguiti per il simbolo durante l'intero ciclo di vita della posizione.	long

Per la funzione HistoryDealGetDouble()

ENUM_DEAL_PROPERTY_DOUBLE

Identificatore	Descrizione	Tipo
DEAL_VOLUME	Volume affare	double
DEAL_PRICE	Prezzo affare	double
DEAL_COMMISSION	Prezzo commissione	double
DEAL_SWAP	Swap cumulativo sulla chiusura	double
DEAL_PROFIT	Profitto affare	double

Identificatore	Descrizione	Tipo
DEAL_FEE	Le commissioni per effettuare un affare vengono addebitate subito dopo aver eseguito l'affare	double
DEAL_SL	Stop Loss level • Entry and reversal deals use the Stop Loss values from the original order based on which the position was opened or reversed • Exit deals use the Stop Loss of a position as at the time of position closing	double
DEAL_TP	Take Profit level • Entry and reversal deals use the Take Profit values from the original order based on which the position was opened or reversed • Exit deals use the Take Profit value of a position as at the time of position closing	double

Per la funzione HistoryDealGetString()

ENUM_DEAL_PROPERTY_STRING

Identificatore	Descrizione	Tipo
DEAL_SYMBOL	Simbolo affare	string
DEAL_COMMENT	Commento affare	string

Ogni affare è caratterizzato da un tipo, i valori consentiti sono elencati in ENUM_DEAL_TYPE. Per ottenere informazioni sul tipo di operazione, utilizzare la funzione HistoryDealGetInteger() con il modificatore DEAL_TYPE.

ENUM_DEAL_TYPE

Identificatore	Descrizione
DEAL_TYPE_BUY	Buy
DEAL_TYPE_SELL	Sell
DEAL_TYPE_BALANCE	Balance
DEAL_TYPE_CREDIT	Credit

Identificatore	Descrizione
DEAL_TYPE_CHARGE	Carico aggiuntivo
DEAL_TYPE_CORRECTION	Correzione
DEAL_TYPE_BONUS	Bonus
DEAL_TYPE_COMMISSION	Commissione aggiuntiva
DEAL_TYPE_COMMISSION_DAILY	Commissione giornaliera
DEAL_TYPE_COMMISSION_MONTHLY	Commissione mensile
DEAL_TYPE_COMMISSION_AGENT_DAILY	Commissione agente giornaliero
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Commissione agente mensile
DEAL_TYPE_INTEREST	Tasso d'interesse
DEAL_TYPE_BUY_CANCELED	Affare buy annullato Non ci può essere una situazione in cui un affare precedentemente eseguito venga annullato. In questo caso, il tipo di affare precedentemente eseguito (DEAL_TYPE_BUY) viene cambiato in DEAL_TYPE_BUY_CANCELED, ed il suo profitto/perdita viene reso a zero. Il precedente profitto/perdita ottenuti vengono caricati/prelevati usando un' operazione di bilanciamento separata.
DEAL_TYPE_SELL_CANCELED	Affare vendita eseguito. Non ci può essere una situazione in cui un affare di vendita eseguito in precedenza venga cancellato. In questo caso, il tipo di affare precedentemente eseguito (DEAL_TYPE_SELL) viene cambiato in DEAL_TYPE_SELL_CANCELED, ed il suo profitto/perdita viene reso a zero. Il precedente profitto/perdita ottenuti vengono caricati/prelevati usando un' operazione di bilanciamento separata.
DEAL_DIVIDEND	Operazioni di dividendo
DEAL_DIVIDEND_FRANKED	Operazioni di dividendi frante (non tassabili)
DEAL_TAX	Addebiti tasse

Gli affari differiscono non solo per i loro tipi impostati in ENUM_DEAL_TYPE, ma anche nel modo in cui cambiano le posizioni. Questa può essere una semplice posizione di apertura, o l'accumulo di una posizione precedentemente aperta (ingresso di mercato), chiusura di posizione da un affare opposto di un volume corrispondente (uscita mercato), o inversone di posizione, se la direzione-opposta dell' affare copre il volume della posizione precedentemente aperta.

Tutte queste situazioni sono descritte da valori dell'enumerazione ENUM_DEAL_ENTRY. Per ricevere queste informazioni su un affare, utilizzare la funzione HistoryDealGetInteger() con il modificatore DEAL_ENTRY.

ENUM_DEAL_ENTRY

Identificatore	Descrizione
DEAL_ENTRY_IN	Entry in
DEAL_ENTRY_OUT	Entry out
DEAL_ENTRY_INOUT	Reverse
DEAL_ENTRY_OUT_BY	Chiude una posizione da una opposta

La ragione dell'esecuzione degli affari è contenuta nella proprietà DEAL_REASON. Un affare può essere eseguito in seguito all'avvio di un ordine piazzato da un'applicazione mobile o da un programma MQL5, nonché come risultato dell'evento StopOut, del calcolo del margine di variazione, ecc. I valori possibili di DEAL_REASON sono descritti nell'enumerazione ENUM_DEAL_REASON. Per gli affari non-di-trading derivanti dal bilancio, credito, commissioni e altre operazioni, DEAL_REASON_CLIENT è indicato come motivo.

ENUM_DEAL_REASON

Identificatore	Descrizione
DEAL_REASON_CLIENT	L'affare è stato eseguita in seguito all'attivazione di un ordine piazzato da un terminale desktop
DEAL_REASON_MOBILE	L'affare è stato eseguito in seguito all'attivazione di un ordine piazzato da un'applicazione mobile
DEAL_REASON_WEB	L'affare è stato eseguito in seguito all'attivazione di un ordine piazzato dalla piattaforma web
DEAL_REASON_EXPERT	L'affare è stato eseguito in seguito all'attivazione di un ordine piazzato da un programma MQL5, ad esempio un Expert Advisor o uno Script
DEAL_REASON_SL	L'affare è stato eseguito in seguito all'attivazione dello Stop Loss
DEAL_REASON_TP	L'affare è stato eseguito in seguito all'attivazione del Take Profit
DEAL_REASON_SO	L'affare è stato eseguito in seguito all'attivazione dell'evento Stop Out
DEAL_REASON_ROLLOVER	L'affare è stato eseguito a causa di un rollover
DEAL_REASON_VMARGIN	L'affare è stato eseguito dopo aver addebitato il margine di variazione
DEAL_REASON_SPLIT	L'affare è stato eseguito dopo la ripartizione (riduzione dei prezzi) di uno strumento, che aveva una posizione aperta durante l'annuncio split

Tipi di Operazioni di Trade

Il trading avviene con l'invio di ordini per aprire posizioni utilizzando la funzione OrderSend(), come pure per piazzare, modificare o cancellare gli ordini pendenti. Ogni ordine di trade si riferisce al tipo di operazione richiesta. Le operazioni di trading sono descritte nella enumerazione ENUM_TRADE_REQUEST_ACTIONS.

ENUM_TRADE_REQUEST_ACTIONS

Identificazione	Descrizione
TRADE_ACTION_DEAL	Piazza un ordine di trade per un' esecuzione immediata con i parametri specificati (ordine di mercato)
TRADE_ACTION_PENDING	Piazza un ordine di trade per l' esecuzione in condizioni specifiche (ordine pendente)
TRADE_ACTION_SLTP	Modifica i valori Stop Loss e Take Profit di una posizione aperta
TRADE_ACTION_MODIFY	Modifica i parametri dell' ordine effettuato in precedenza
TRADE_ACTION_REMOVE	Elimina l'ordine pendnete piazzato in precedenza
TRADE_ACTION_CLOSE_BY	Chiude una posizione da una opposta

Esempio per TRADE_ACTION_DEAL, operazione di trade per aprire una posizione Buy:

```
#define EXPERT_MAGIC 123456   // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Apre posizione Buy                                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parametri della richiesta
   request.action   =TRADE_ACTION_DEAL;                     // tipo di operazione di t
   request.symbol   =Symbol();                              // simbolo
   request.volume   =0.1;                                   // volume di 0.1 lotti
   request.type     =ORDER_TYPE_BUY;                        // tipo di ordine
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // prezzo per l'apertura
   request.deviation=5;                                     // permessa la deviazione
   request.magic    =EXPERT_MAGIC;                          // MagicNumber dell'ordine
//--- invia la richiesta
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se impossibilitato ad i
//--- informazione riguardo l'operazione
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Esempio di TRADE_ACTION_DEAL, operazione di trade per aprire la posizione Sell:

```
#define EXPERT_MAGIC 123456   // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Apre posizione Sell                                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parametri della richiesta
   request.action   =TRADE_ACTION_DEAL;                     // tipo di operazione di t
   request.symbol   =Symbol();                              // simbolo
   request.volume   =0.2;                                   // volume di 0.2 lotti
   request.type     =ORDER_TYPE_SELL;                       // tipo di ordine
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // prezzo per l'apertura
   request.deviation=5;                                     // permessa la deviazione
   request.magic    =EXPERT_MAGIC;                          // MagicNumber dell'ordine
//--- invia la richiesta
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se impossibilitato ad i
//--- informazione riguardo l'operazione
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Esempio di TRADE_ACTION_DEAL, operazione di trade per chiudere la posizione:

```
#define EXPERT_MAGIC 123456   // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Chiude tutte le posizioni                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // numero della posizione aperta
//--- itera su tutte le posizioni
   for(int i=total-1; i>=0; i--)
     {
      //--- parametri per l'ordine
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- informazioni output riguardo la posizione
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- se il MagicNumber corrisponde
      if(magic==EXPERT_MAGIC)
        {
         //--- azzera la richiesta ed i valori risultato
         ZeroMemory(request);
         ZeroMemory(result);
         //--- imposta i parametri dell'operazione
         request.action   =TRADE_ACTION_DEAL;        // tipo di operazione di trade
         request.position =position_ticket;          // ticket della posizione
         request.symbol   =position_symbol;          // simbolo
         request.volume   =volume;                   // volume della posizione
         request.deviation=5;                        // deviazione consentita dal prez
         request.magic    =EXPERT_MAGIC;             // MagicNumber della posizione
         //--- imposta il prezzo ed il tipo d'ordine a seconda del tipo della posizion
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- informazioni output riguardo la chiusura
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToString
         //--- imposta la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend errore %d",GetLastError());  // se impossibile invi
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

```
      }
   }
//+------------------------------------------------------------------+
```

Esempio di TRADE_ACTION_PENDING, operazione di trade per piazzare un ordine pendente:

```
#property description "Esempio di piazzamento dell'ordine pendente"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // MagicNumber dell'expert
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;    // tipo d'ordine
//+------------------------------------------------------------------+
//| Piazza l'ordine pendente                                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parametri per piazzare un ordine pendente
   request.action   =TRADE_ACTION_PENDING;                          // tipo di oper
   request.symbol   =Symbol();                                      // simbolo
   request.volume   =0.1;                                           // volume di 0.
   request.deviation=2;                                             // ammonetare d
   request.magic    =EXPERT_MAGIC;                                  // MagicNumber
   int offset = 50;                                                 // offset from
   double price;                                                    // prezzo d'inn
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);             // valore del p
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);             // numero di ci
   //--- controllo del tipo d'operazione
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                      // tipo d'ordin
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // prezzo per l
      request.price    =NormalizeDouble(price,digits);             // prezzo d'ape
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                      // tipo d'ordi
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;     // prezzo per
      request.price    =NormalizeDouble(price,digits);              // prezzo d'ap
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                           // tipo d'ordi
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // prezzo per
      request.price=NormalizeDouble(price,digits);                 // prezzo d'ap
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                       // tipo d'ordi
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;     // prezzo per
      request.price    =NormalizeDouble(price,digits);              // prezzo d'ap
     }
   else Alert("Questo esempio è solo per piazzare ordini pendenti");   // se non ci so
//--- invia la richiesta
   if(!OrderSend(request,result))
      PrintFormat("OrderSend errore %d",GetLastError());               // se impossi
//--- informazioni riguardo l'operazione
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Esempio per TRADE_ACTION_SLTP, operazione di trade per la modifica dei valori di Stop Loss e Take Profit di una posizione aperta:

```
#define EXPERT_MAGIC 123456  // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Modifica dello Stop Loss e Take Profit della posizione           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // numero delle posizioni aperte
//--- itera su tutte le posizioni/t34>
   for(int i=0; i<total; i++)
     {
      //--- parametri dell'ordine
      ulong  position_ticket=PositionGetTicket(i);// ticket della posizione
      string position_symbol=PositionGetString(POSITION_SYMBOL); // simbolo
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // numero d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber della posizione
      double volume=PositionGetDouble(POSITION_VOLUME);    // volume della posizione
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss della posizione
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit della posizione
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- output informazioni riguardo la posizione
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se il MagicNumber corrisponde, Stop Loss e Take Profit non sono stati defi
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

```
         //--- calcola i livelli del prezzo corrente
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_L
         double price_level;
         //--- se l'offset distanza minima consentita in punti dal corrente punto di c
         if(stop_level<=0)
            stop_level=150; // imposta l'offset distanza di 150 punti dal corrente pun
         else
            stop_level+=50; // imposta l'offset distanza a (SYMBOL_TRADE_STOPS_LEVEL +

         //--- calcolo ed arrotondamento dei valori Stop Loss e Take Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(ask+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(bid-price_level,digits);
           }
         //--- azzero la richiesta ed i risultanti valori
         ZeroMemory(request);
         ZeroMemory(result);
         //--- setting the operation parameters
         request.action  =TRADE_ACTION_SLTP; // type of trade operation
         request.position=position_ticket;   // ticket of the position
         request.symbol=position_symbol;     // simbolo
         request.sl      =sl;                // Stop Loss della posizione
         request.tp      =tp;                // Take Profit della posizione
         request.magic=EXPERT_MAGIC;         // MagicNumber della posizione
         //--- output informazioni riguardo la modifica
         PrintFormat("Modifica #%I64d %s %s",position_ticket,position_symbol,EnumToStr
         //--- invia la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // se impossibile invia
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Esempio per TRADE_ACTION_MODIFY, operazione di trade per la modifica dei livelli di prezzo degli ordini pendenti:

```
#define EXPERT_MAGIC 123456  // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Modifica degli ordini pendenti                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // numero totale di orini pendenti piazzati
//--- itera su tutti gli ordini pendenti piazzati
   for(int i=0; i<total; i++)
     {
      //--- parametri dell'ordine
      ulong  order_ticket=OrderGetTicket(i);                          // order tick
      string order_symbol=Symbol();                                   // simbolo
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // numero di
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // MagicNumbe
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // corrente v
      double sl=OrderGetDouble(ORDER_SL);                             // corrente S
      double tp=OrderGetDouble(ORDER_TP);                             // corrente T
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // tipo di or
      int offset = 50;                                                // offset dal
      double price;                                                   // prezzo d'i
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // valore in
      //--- output informazioni riguardo l'ordine
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se il MagicNumber corrisponde, Stop Loss e Take Profit non sono definiti
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                          // tipo d'opera
         request.order = OrderGetTicket(i);                           // ticket ordi
         request.symbol   =Symbol();                                  // simbolo
         request.deviation=5;                                         // deviazione
        //--- imposta il livello del prezzo, Take Profit e Stop Loss dell'ordine diper
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prezzo
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prezzo
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // prezzo
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // prezzo
           }
         //--- invia la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend errore %d",GetLastError());  // se impossibile invi
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- azzera la richiesta ed i risultanti valori
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Esempio per TRADE_ACTION_REMOVE, operazione di trade per eliminare ordini pendenti:

```
#define EXPERT_MAGIC 123456  // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Elimina ordini pendenti                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // numero totale di ordini pendenti piazzati
//--- itera su tutti gli ordini pendenti piazzati
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // ticket ordine
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // MagicNumber dell'ord
      //--- se il MagicNumber corrisponde
      if(magic==EXPERT_MAGIC)
        {
         //--- azzera la richiesta ed i valori del risultato
         ZeroMemory(request);
         ZeroMemory(result);
         //--- imposta i parametri dell'operazione
         request.action=TRADE_ACTION_REMOVE;                   // tipo d'operazione di
         request.order = order_ticket;                         // ticket ordine
         //--- invia la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend errore %d",GetLastError());  // se impossibile invi
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Esempio per TRADE_ACTION_CLOSE_BY, operazione di trade per chiudere posizioni da posizioni opposte:

```
#define EXPERT_MAGIC 123456  // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Chiude tutte le posizioni per posizioni opposte                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiare ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // numero di posizioni aperte
//--- itera su tutte le posizioni
   for(int i=total-1; i>=0; i--)
     {
      //--- parametri dell'ordine
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      double sl=PositionGetDouble(POSITION_SL);
      double tp=PositionGetDouble(POSITION_TP);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- output informazioni riguardo la posizione
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se il MagicNumber corrisponde
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // simbolo della posizione opposta
            //--- se i simboli delle posizioni opposte ed iniziali corrispondono
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC))
              {
               //--- imposta il topo di posizione opposta
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)PositionGetInteger(POSIT
               //--- lascia, se il tipo di posizione iniziale ed opposta corrispondono
               if(type==type_by)
                  continue;
               //--- azzera la richiesta ed i risulanti valori
               ZeroMemory(request);
               ZeroMemory(result);
               //--- imposta i parametri dell'operazione
               request.action=TRADE_ACTION_CLOSE_BY;                    // tipo d
               request.position=position_ticket;                        // ticket
               request.position_by=PositionGetInteger(POSITION_TICKET); // ticket
               //request.symbol     =position_symbol;
               request.magic=EXPERT_MAGIC;                              // MagicN
               //--- output informazioni riguardo la chiusura per posizione opposta
               PrintFormat("Chiudi#%I64d %s %s da #%I64d",position_ticket,position_sym
               //--- invia la richiesta
               if(!OrderSend(request,result))
                  PrintFormat("OrderSend errore %d",GetLastError()); // se impossibile
```

```
                //--- informazioni riguardo l'operazione
                PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result
               }
            }
         }
      }
   }
```

Tipi di Transazioni di Trade

Quando si eseguono alcune azioni precise su un trade account, il suo stato cambia. Tali azioni comprendono:

- Inviare una richiesta di trade da qualsiasi applicazione MQL5 nel terminale client utilizzando le funzioni OrderSend e OrderSendAsync e la sua ulteriore esecuzione;
- Inviare una richiesta di trade tramite l'interfaccia grafica del terminale e la sua esecuzione ulteriore;
- Attivazione di ordini pendenti ed ordini di stop sul server;
- Esecuzione di operazioni sul lato trade server.

Le operazioni commerciali di seguito vengono eseguite come risultato di queste azioni:

- gestione di una richiesta di trade;
- cambio di ordini aperti;
- cambio della cronistoria degli ordini;
- cambio della cronistoria degli affari;
- cambio delle posizioni.

Per esempio, quando si invia un ordine di acquisto di mercato, esso viene gestito, un ordine di acquisto appropriata viene creato per l'account, l'ordine quindi viene eseguito e rimosso dalla lista di quelli aperti, e poi viene aggiunto alla cronistoria ordini, un appropriato aff viene aggiunto alla cronistoria ed una nuova posizione viene creata. Tutte queste azioni sono transazioni di trade.

Per lasciare che un programmatore possa monitorare le azioni eseguite in relazione a un account di trade, viene fornita la funzione OnTradeTransaction . Questo handler permette di ottenere transazioni di trade applicate ad un account in applicazioni MQL5. La descrizione della transazione di trade è presentata nel primo parametro di OnTradeTransaction utilizzando la struttura MqlTradeTransaction.

Il tipo di transazione di trade viene immesso nel parametro tipo della struttura MqlTradeTransaction. Possibili tipi di transazioni di trade vengono descritte dall'enumerazione seguente:

ENUM_TRADE_TRANSACTION_TYPE

Identificatore	Descrizione
TRADE_TRANSACTION_ORDER_ADD	L'aggiunta di un nuovo ordine aperto.
TRADE_TRANSACTION_ORDER_UPDATE	Aggiornamento di un ordine aperto. Gli aggiornamenti includono non solo le evidenti modifiche dal terminale client o dal lato trade server, ma anche cambiamenti dello stato di un ordine quando lo si imposta (ad esempio, passaggio da ORDER_STATE_STARTED a ORDER_STATE_PLACED o da ORDER_STATE_PLACED a ORDER_STATE_PARTIAL, ecc).
TRADE_TRANSACTION_ORDER_DELETE	La rimozione di un ordine dalla lista di quelli aperti. Un ordine può essere eliminato da quelli aperti a seguito dell' impostazione di un' appropriata richiesta o esecuzione (riempimento) e spostandosi nella cronistoria.

Identificatore	Descrizione
TRADE_TRANSACTION_DEAL_ADD	Aggiunta di un affare per la cronistoria. L'azione viene eseguita come risultato di un' esecuzione di ordine o l'esecuzione di operazioni su un saldo del conto.
TRADE_TRANSACTION_DEAL_UPDATE	Aggiornamento di un affare nella cronistoria. Ci possono essere casi in cui un accordo applicato in precedenza viene modificato su un server. Per esempio, un accordo è stato cambiato in un sistema commerciale esterno (scambio) in cui era precedentemente trasferito da un broker.
TRADE_TRANSACTION_DEAL_DELETE	Eliminazione di un affare dalla cronistoria. Ci possono essere casi in cui un accordo precedentemente eseguito viene eliminato da un server. Per esempio, un accordo è stato eliminato in un sistema di trade esterno (scambio) in cui era precedentemente trasferito da un broker.
TRADE_TRANSACTION_HISTORY_ADD	Aggiunta di un ordine allo storico come risultato di esecuzione o cancellazione.
TRADE_TRANSACTION_HISTORY_UPDATE	Modifica un ordine situato nella storia ordini. Questo tipo viene fornito per migliorare la funzionalità sul lato trade server.
TRADE_TRANSACTION_HISTORY_DELETE	Deleting an order from the orders history. Questo tipo viene fornito per migliorare la funzionalità sul lato trade server.
TRADE_TRANSACTION_POSITION	Modifica di una posizione non correlata ad un'esecuzione di affare. Questo tipo di transazione mostra che una posizione è stata modificata dal lato trade server. Volume della posizione, prezzo di apertura, Stop Loss e Take Profit possono essere modificati. I dati sulle modifiche vengono inviati nella struttura MqlTradeTransaction tramite l'handler OnTradeTransaction. La modifica della posizione (aggiunta, modifica o chiusura), come risultato dell'esecuzione di un affare, non porta al verificarsi della transazione TRADE_TRANSACTION_POSITION.
TRADE_TRANSACTION_REQUEST	La notifica del fatto che una richiesta di trade sia stata elaborata dal server e ed il risultato dell'elaborazione sia stato ricevuto. Solo il campo tipo (tipo di transazione di trade) deve essere analizzato per tali operazioni nella struttura MqlTradeTransaction. Il secondo e terzo parametro di OnTradeTransaction (richiesta e risultato) devono essere analizzati per ulteriori dati.

A seconda del tipo di transazione di trade, vari parametri vengono riempiti nella struttura MqlTradeTransaction descrivendola. Una descrizione dettagliata dei dati presentati è mostrata in "Struttura di una transazione di Trade".

Vedi anche

Strutture di una Transazione di Trade, OnTradeTransaction

Ordini di Trade nel Depth Of Market (_* profondità di mercato)

Per i titoli di capitale, il Depth of Market è disponibile, dove si possono vedere gli ordini Buy e Sell in corso. La direzione desiderata di un'operazione di trade, l'importo richiesto e prezzo richiesto sono specificati per ogni ordine.

Per ottenere informazioni sullo stato corrente del DOM dal mezzo MQL5, vengono utilizzate le funzioni MarketBookGet(), che pongono la "schermata" DOM nell'array di strutture MqlBookInfo. Ciascun elemento della matrice nel campo *field* contiene informazioni sulla direzione dell 'ordine - il valore dell'enumerazione ENUM_BOOK_TYPE.

ENUM_BOOK_TYPE

Identificatore	Descrizione
BOOK_TYPE_SELL	Ordine di Sell(Offerta)
BOOK_TYPE_BUY	Ordine di Buy (Acquisto)
BOOK_TYPE_SELL_MARKET	Ordine di Sell dal Mercato
BOOK_TYPE_BUY_MARKET	Ordine di Buy dal Mercato

Vedi anche

Le strutture e le classi, Struttura del DOM, Tipi di operazioni di Trade, Informazioni di Mercato

Proprietà dei Segnali

Le seguenti enumerazioni vengono utilizzate quando si lavora con segnali di trading e le impostazioni di copia del segnale.

Enumerazione di proprietà di tipo double dei segnali di trading:

ENUM_SIGNAL_BASE_DOUBLE

ID	Descrizione
SIGNAL_BASE_BALANCE	Account: bilancio
SIGNAL_BASE_EQUITY	Account: equità
SIGNAL_BASE_GAIN	Account: guadagno
SIGNAL_BASE_MAX_DRAWDOWN	Account: massimo drawdown
SIGNAL_BASE_PRICE	Prezzo di sottoscrizione al Segnale
SIGNAL_BASE_ROI	Ritorno dell' Investimento(%)

Enumerazione di proprietà di tipo integer dei segnali di trading:

ENUM_SIGNAL_BASE_INTEGER

ID	Descrizione
SIGNAL_BASE_DATE_PUBLISHED	Data di pubblicazione (la data di quando è diventato disponibile per la sottoscrizione)
SIGNAL_BASE_DATE_STARTED	Data di inizio monitoraggio
SIGNAL_BASE_DATE_UPDATED	Data dell'ultimo aggiornamento delle statistiche di trading del segnale
SIGNAL_BASE_ID	ID del Segnale
SIGNAL_BASE_LEVERAGE	Leveraggio dell'account
SIGNAL_BASE_PIPS	Prifitti in pips
SIGNAL_BASE_RATING	Posizione nel rating
SIGNAL_BASE_SUBSCRIBERS	Numero di sottoscritti
SIGNAL_BASE_TRADES	Numero di trades
SIGNAL_BASE_TRADE_MODE	Tipo di account (0-reale, 1-demo, 2-contest)

Enumerazione di proprietà di tipo string dei segnali di trading:

ENUM_SIGNAL_BASE_STRING

ID	Descrizione
SIGNAL_BASE_AUTHOR_LOGIN	Login Autore

ID	Descrizione
SIGNAL_BASE_BROKER	Nome del Broker (società)
SIGNAL_BASE_BROKER_SERVER	Server del Broker
SIGNAL_BASE_NAME	Nome del segnale
SIGNAL_BASE_CURRENCY	Signal base currency

Enumerazione di proprietà di tipo integer delle impostazioni di copia dei segnali:

ENUM_SIGNAL_INFO_INTEGER

ID	Descrizione
SIGNAL_INFO_CONFIRMATIONS_DISABLED	The flag enables synchronization without confirmation dialog
SIGNAL_INFO_COPY_SLTP	Flag di Copia Stop Loss e Take Profit
SIGNAL_INFO_DEPOSIT_PERCENT	Deposito in percentuale (%)
SIGNAL_INFO_ID	Il Signal id
SIGNAL_INFO_SUBSCRIPTION_ENABLED	"Copy trades by subscription" permission flag
SIGNAL_INFO_TERMS_AGREE	Flag "Acconsento ai termini di uso del servizio Segnali"

Enumerazione di proprietà di tipo double delle impostazioni di copia dei segnali:

ENUM_SIGNAL_INFO_DOUBLE

ID	Descrizione
SIGNAL_INFO_EQUITY_LIMIT	Limite equità
SIGNAL_INFO_SLIPPAGE	Lo slippage (utilizzato quando si posizionano ordini di mercato nella sincronizzazione di posizioni e la copia dei trades)
SIGNAL_INFO_VOLUME_PERCENT	Massims percentuale di utiilizzo del deposito (%)

Enumerazione di proprietà di tipo <li1>string</li1><t2> delle impostazioni di copia dei segnali:</t2>

ENUM_SIGNAL_INFO_STRING

ID	Descrizione
SIGNAL_INFO_NAME	Nome del segnale

Vedi anche

Segnali di Trade

Named Constants

Tutte le costanti utilizzate in MQL5 possono essere suddivise nei seguenti gruppi:

- Sostituzioni macro predefinite - i valori vengono sostituiti durante la compilazione;
- Costanti matematiche - valori di alcune espressioni matematiche;
- Costanti di tipo numerico - alcune delle restrizioni di tipo semplice;
- Codici di motivo deinizializzazione - Descrizione dei motivi di deinizializzazione;
- Controllo Puntatore Oggetto - enumerazione dei tipi di puntatori restituita dalla funzione CheckPointer();
- Altre costanti - tutte le altre costanti.

Sostituzioni Macro predefinite

Per semplificare il processo di debug e ottenere informazioni sul funzionamento di un programma-MQL5, ci sono macro costanti speciali, i cui valori sono stabiliti al momento della compilazione. Il modo più semplice per utilizzare queste costanti è l'output di valori dalla funzione Print(), come è illustrato nell'esempio.

Constant	Descrizione
__CPU_ARCHITECTURE__	Nome dell'architettura (set di comandi) per il quale il file EX5 viene compilato
__DATE__	Compilazione di file senza orario (ore, minuti e secondi sono uguali a 0)
__DATETIME__	Data ed ora di compilazione del file
__LINE__	Numero di riga nel codice sorgente, in cui si trova la macro
__FILE__	Nome del file attualmente elaborati
__PATH__	Un percorso assoluto del file che è attualmente in fase di compilazione
__FUNCTION__	Nome della funzione, nel cui corpo si trova la macro
__FUNCSIG__	Firma della funzione nel cui corpo si trova la macro. Logging della descrizione completa delle funzioni può essere utile per l'identificazione di funzioni sovraccaricate
__MQLBUILD__, __MQL5BUILD__	Numero di build compilatore
__COUNTER__	The compiler for each encountered __COUNTER__ declaration substitutes the counter value from 0 to N-1 where N is a number of uses in the code. The __COUNTER__ order is guaranteed when recompiling the source code with no changes. The __COUNTER__ value is calculated the following way: • the initial counter value is 0, • after each counter usage, its value is increased by 1, • first, the compiler expands all macros and templates into source code on-site, • a separate code is created for each template function specialization, • a separate code is created for each template class/structure specialization, • next, the compiler passes through the obtained source code in the defined order and replaces each detected __COUNTER__ usage with the current counter value. The example below shows how the compiler handles the source code and replaces all instances of __COUNTER__ it meets with sequentially increasing values.

Constant	Descrizione
__RANDOM__	The compiler inserts a random ulong value for each __RANDOM__ declaration.

Esempio:

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                        |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- un esempio di output dati in fase di inizializzazione Expert Advisor
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- impostare l'intervallo tra gli eventi timer
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- un esempio di informazioni output alla deinizializzazione Expert Advisor
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//| Funzione tick dell'Expert                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- output informazioni al ricevimento tick
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//| test1                                                            |
//+------------------------------------------------------------------+
void test1(string par)
  {
//--- output informazioni all'interno della funzione
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
  }
//+------------------------------------------------------------------+
```

```
//| test2                                                                                |
//+--------------------------------------------------------------------------------------+
void test2()
  {
//--- output informazioni all'interno della funzione
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+--------------------------------------------------------------------------------------+
//| OnTimer event handler                                                                |
//+--------------------------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

The example for learning how to work with the __COUNTER__ macro

```
//--- create a macro for a quick display of the expression and its value in the journa
#define print(expr) Print(#expr,"=",expr)

//--- define the MACRO_COUNTER custom macro via the predefined __COUNTER__ macro
#define MACRO_COUNTER __COUNTER__

//--- set the input value of the variable using the __COUNTER__ macro
input int InpVariable = __COUNTER__;

//--- set the value of the global variable using the __COUNTER__ macro before defining
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| the function returns the __COUNTER__ value                       |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| the template function returns the __COUNTER__ value              |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| the structure with the method returning __COUNTER__              |
```

```
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // not used

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| the template structure with the method returning __COUNTER__     |
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // not used

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| the structure with the template method returning __COUNTER__     |
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // not used

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| the function #2, which returns the __COUNTER__ value             |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
// __COUNTER__ in the macro and the variables
```

```
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);

//--- __COUNTER__ in the functions
   print(GlobalFunc());
   print(GlobalFunc());                 // the value is not changed
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());    // the value is not changed
   print(GlobalTemplateFunc<double>());// the value has changed
   print(GlobalFunc2());
   print(GlobalFunc2());                // the value is not changed

// __COUNTER__ in the structure
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                  // the value is not changed

// __COUNTER__ in the template structure
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                  // the value is not changed
   print(b3.Method());                  // the value has changed

// __COUNTER__ in the structure with the template function
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());          // the value has changed
   print(c2.Method<int>());             // the same value as during the first c1.Metho

//--- let's have another look at __COUNTER__ in the macro and the global variable
   print(MACRO_COUNTER);  // the value has changed
   print(ExtGlobal2);
  }
//--- set the value of the global variable using the __COUNTER__ macro after defining
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* Result
   __COUNTER__=3
   InpVariable=0
   ExtVariable=1
   GlobalFunc()=5
   GlobalFunc()=5
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<double>()=9
   GlobalFunc2()=7
```

```
   GlobalFunc2()=7
   a1.Method()=6
   a2.Method()=6
   b1.Method()=10
   b2.Method()=10
   b3.Method()=11
   c1.Method<int>()=12
   c1.Method<double>()=13
   c2.Method<int>()=12
   __COUNTER__=4
   ExtGlobal2=2

*/
```

Mathematical Constants

Costanti speciali contenenti valori sono riservate per alcune espressioni matematiche. Queste costanti possono essere utilizzate in qualsiasi luogo del programma invece di calcolare i valori usando funzioni matematiche.

Constant	Descrizione	Valore
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.43429448190325182765 1
M_LN2	ln(2)	0.69314718055994530941 7
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.78539816339744830962
M_1_PI	1/pi	0.31830988618379067153 8
M_2_PI	2/pi	0.63661977236758134307 6
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.70710678118654752440 1

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- stampa i valori delle costanti
   Print("M_E = ",DoubleToString(M_E,16));
   Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
   Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
   Print("M_LN2 = ",DoubleToString(M_LN2,16));
   Print("M_LN10 = ",DoubleToString(M_LN10,16));
   Print("M_PI = ",DoubleToString(M_PI,16));
   Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
   Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
   Print("M_1_PI = ",DoubleToString(M_1_PI,16));
   Print("M_2_PI = ",DoubleToString(M_2_PI,16));
   Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
   Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
   Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
```

```
}
```

Costanti di tipo numerico

Ogni tipo semplice numerico è previsto per un certo tipo di compiti e consente di ottimizzare il funzionamento di un programma-mql5 se utilizzato correttamente. Per una migliore leggibilità del codice e la corretta gestione dei risultati di calcolo, ci sono delle costanti che permettono di ricevere informazioni sulle restrizioni per un certo tipo di dati semplici.

Constant	Descrizione	Valore
CHAR_MIN	Valore minimo, che può essere rappresentato dal tipo char	-128
CHAR_MAX	Valore massimo, che può essere rappresentato dal tipo char	127
UCHAR_MAX	Valore massimo, che può essere rappresentato dal tipo uchar	255
SHORT_MIN	Valore minimo, che può essere rappresentato dal tipo short	-32768
SHORT_MAX	Valore massimo, che può essere rappresentato dal tipo short	32767
USHORT_MAX	Valore massimo, che può essere rappresentato dal tipo ushort	65535
INT_MIN	Valore minimo, che può essere rappresentato dal tipo int	-2147483648
INT_MAX	Valore massimo, che può essere rappresentato dal tipo int	2147483647
UINT_MAX	Valore massimo, che può essere rappresentato dal tipo unit	4294967295
LONG_MIN	Valore minimo, che può essere rappresentato dal tipo long	-9223372036854775808
LONG_MAX	Valore massimo, che può essere rappresentato dal tipo long	9223372036854775807
ULONG_MAX	Valore massimo, che può essere rappresentato dal tipo ulong	18446744073709551615
DBL_MIN	Valore minimo, che può essere rappresentato dal tipo double	2.2250738585072014e-308
DBL_MAX	Valore massimo, che può essere rappresentato dal tipo double	1.7976931348623158e+308
DBL_EPSILON	Valore minimo, che soddisfa la condizione: 1.0+DBL_EPSILON != 1.0 (per il tipo double)	2.2204460492503131e-016
DBL_DIG	Numero di cifre decimali significative per il tipo double	15

Constant	Descrizione	Valore
DBL_MANT_DIG	Conteggio bits in una mantissa per il tipo double	53
DBL_MAX_10_EXP	Massimo valore decimale del grado dell' esponente per il tipo double	308
DBL_MAX_EXP	Massimo valore binario del grado dell' esponente per il tipo double	1024
DBL_MIN_10_EXP	Minimo valore decimale del grado dell' esponente per il tipo double	(-307)
DBL_MIN_EXP	Minimo valore binario del grado dell' esponente per il tipo double	(-1021)
FLT_MIN	Minimo valore positivo, che può essere rappresentato dal tipo float	1.175494351e-38
FLT_MAX	Valore massimo, che può essere rappresentato dal tipo float	3.402823466e+38
FLT_EPSILON	Valore minimo, che soddisfa la condizione: 1.0+DBL_EPSILON != 1.0 (per il tipo float)	1.192092896e-07
FLT_DIG	Numero di cifre decimali significative per il tipo float	6
FLT_MANT_DIG	Conteggio Bits nella mantissa per il tipo float	24
FLT_MAX_10_EXP	Massimo valore decimale del grado dell'esponente per il tipo float	38
FLT_MAX_EXP	Valore binario massimo dell' esponente del grado dell'esponente per il tipo float	128
FLT_MIN_10_EXP	Valore decimale minimo del grado dell'esponente per il tipo float	-37
FLT_MIN_EXP	Valore binario minimo del grado dell' esponente per il tipo float	(-125)

Esempio:

```
voidOnStart()
  {
//--- stampa i valori costanti
   printf("CHAR_MIN = %d",CHAR_MIN);
   printf("CHAR_MAX = %d",CHAR_MAX);
   printf("UCHAR_MAX = %d",UCHAR_MAX);
   printf("SHORT_MIN = %d",SHORT_MIN);
   printf("SHORT_MAX = %d",SHORT_MAX);
   printf("USHORT_MAX = %d",USHORT_MAX);
   printf("INT_MIN = %d",INT_MIN);
```

```
   printf("INT_MAX = %d",INT_MAX);
   printf("UINT_MAX = %u",UINT_MAX);
   printf("LONG_MIN = %I64d",LONG_MIN);
   printf("LONG_MAX = %I64d",LONG_MAX);
   printf("ULONG_MAX = %I64u",ULONG_MAX);
   printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
   printf("DBL_MIN = %.16e",DBL_MIN);
   printf("DBL_MAX = %.16e",DBL_MAX);
   printf("DBL_EPSILON = %.16e",DBL_EPSILON);
   printf("DBL_DIG = %d",DBL_DIG);
   printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
   printf("DBL_MAX_10_EXP =  %d",DBL_MAX_10_EXP);
   printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
   printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
   printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
   printf("FLT_MIN = %.8e",FLT_MIN);
   printf("FLT_MAX = %.8e",FLT_MAX);
   printf("FLT_EPSILON = %.8e",FLT_EPSILON);
/*
   CHAR_MIN = -128
   CHAR_MAX = 127
   UCHAR_MAX = 255
   SHORT_MIN = -32768
   SHORT_MAX = 32767
   USHORT_MAX = 65535
   INT_MIN = -2147483648
   INT_MAX = 2147483647
   UINT_MAX = 4294967295
   LONG_MIN = -9223372036854775808
   LONG_MAX = 9223372036854775807
   ULONG_MAX = 18446744073709551615
   EMPTY_VALUE = 1.7976931348623157e+308
   DBL_MIN = 2.2250738585072014e-308
   DBL_MAX = 1.7976931348623157e+308
   DBL_EPSILON = 2.2204460492503131e-16
   DBL_DIG = 15
   DBL_MANT_DIG = 53
   DBL_MAX_10_EXP =  308
   DBL_MAX_EXP = 1024
   DBL_MIN_10_EXP = -307
   DBL_MIN_EXP = -1021
   FLT_MIN = 1.17549435e-38
   FLT_MAX = 3.40282347e+38
   FLT_EPSILON = 1.19209290e-07
*/
  }
```

Codici del Motivo di DeInizializzazione

Motivo della deinizializzazione (codici) vengono restituiti dalla funzione UninitializeReason(). I valori possibili sono i seguenti:

Constant	Valore	Descrizione
REASON_PROGRAM	0	Expert Advisor terminata la loro operazione chiamando la funzione ExpertRemove()
REASON_REMOVE	1	Il programma è stato eliminato dal grafico
REASON_RECOMPILE	2	Il programma è stato ricompilato
REASON_CHARTCHANGE	3	Il simbolo o un periodo del grafico è stato cambiato
REASON_CHARTCLOSE	4	Il grafico è stato chiuso
REASON_PARAMETERS	5	I parametri di input sono stati modificati dall' utente
REASON_ACCOUNT	6	Un altro account è stato attivato o si è verificata la riconnessione al trade server a causa di cambiamenti nelle impostazioni dell'account
REASON_TEMPLATE	7	Un nuovo template è stato applicato
REASON_INITFAILED	8	Questo valore indica che l'handler OnInit() ha restituito un valore diverso da zero
REASON_CLOSE	9	Il terminale è stato chiuso

Il codice motivo di deinizializzazione è anche passato come parametro della funzione predeterminata OnDeinit(motivo const int).

Esempio:

```
//+------------------------------------------------------------------+
//| ottiene la descrizione del testo                                 |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="L'account è cambiato";break;
      case REASON_CHARTCHANGE:
         text="Il simbolo o timeframe sono cambiati";break;
      case REASON_CHARTCLOSE:
         text="Il grafico è cambiato";break;
      case REASON_PARAMETERS:
         text="I parametri di input sono cambiati";break;
      case REASON_RECOMPILE:
```

```
         text="Il programma "+__FILE__+" è stato ricompilato";break;
      case REASON_REMOVE:
         text="Il programma"+__FILE__+" è stato rimosso dal grafico";break;
      case REASON_TEMPLATE:
         text="Nuovi template sono stati applicati al grafico";break;
      default:text="Altro motivo";
     }
//---
   return text;
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                                                 |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Il primo modo per ottenere il codice del motivo di deinizializzazione
   Print(__FUNCTION__,"_Deinizializzazione(codice del motivo) = ",reason);
    //--- Il secondo modo per ottenere il codice del motivo di deinizializzazione
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

Controllo del Puntatore Oggetti

La funzione CheckPointer() viene utilizzata per controllare il tipo di oggetto puntatore. La funzione restituisce un valore dell'enumerazione ENUM_POINTER_TYPE. Se un puntatore non corretto viene utilizzato, l'esecuzione del programma sarà immediatamente terminata.

Oggetti creati dall'operatore new() sono di tipo POINTER_DYNAMIC. L' operatore delete() può e deve essere utilizzato solo per puntatori del genere.

Tutti gli altri puntatori sono di tipo POINTER_AUTOMATIC, il che significa che l'oggetto è stato creato automaticamente dall'ambiente programma mql5. Tali oggetti vengono eliminati automaticamente dopo l'uso.

ENUM_POINTER_TYPE

Constant	Descrizione
POINTER_INVALID	Puntatore errato
POINTER_DYNAMIC	Puntatore dell'oggetto creato dall'operatore new()
POINTER_AUTOMATIC	Puntatore di tutti gli oggetti creati automaticamente (non utilizzando new())

Vedi anche

Errori di runtime, Oggetto Elimina Operatore delete, CheckPointer

Altre costanti

La costante CLR_NONE viene utilizzata per delineare l'assenza di colore, significa che l' oggetto grafico o serie grafica di un indicatore non verrà stampata. Questa costante non è stata inclusa nella lista costanti Web-color, ma può essere applicata ovunque dove gli argomenti del colore sono richiesti.

La costante INVALID_HANDLE può essere utilizzata per il controllo degli handles file (vedi FileOpen() e FileFindFirst()).

Constant	Descrizione	Valore
CHARTS_MAX	Il numero massimo possibile di grafici aperti simultaneamente nel terminale	100
clrNONE	Assenza di colore	-1
EMPTY_VALUE	Valore vuoto in un buffer di indicatore	DBL_MAX
INVALID_HANDLE	Incorrect handle	-1
IS_DEBUG_MODE	Flag dove un programma-mq5 opera in modalità di debug	diverso da zero in modalità di debug, altrimenti zero
IS_PROFILE_MODE	Flag dove un programma-mq5 opera in modalità di analisi	diverso da zero in modalità di analisi, altrimenti zero
NULL	Zero per qualsiasi tipo	0
WHOLE_ARRAY	Indica il numero di elementi restanti fino alla fine dell'array, cioè, verrà elabolrato l'intero array	-1
WRONG_VALUE	La costante può essere implicitamente lanciata a qualsiasi tipo di enumerazione	-1

La costante EMPTY_VALUE di solito corrisponde ai valori di indicatori che non sono mostrati nel grafico. Per esempio, per indicatori built-in Deviazione standard con un periodo di 20, la linea per le prime 19 barre nello storico non è mostrata nel grafico. Se si crea un handle di questo indicatore con la funzione iStdDev() e lo si copia in un array di valori degli indicatori per queste barre attraverso CopyBuffer(), allora questi valori saranno uguali ad EMPTY_VALUE.

È possibile scegliere di specificare per un indicatore personalizzato il proprio valore vuoto dell'indicatore, quando l'indicatore non deve essere compilato nel grafico. Utilizzando la funzione PlotIndexSetDouble() con il modificatore PLOT_EMPTY_VALUE.

La costante NULL può essere assegnata ad una variabile di qualsiasi tipo semplice o ad una struttura oggetto o puntatore alla classe. L'assegnazione NULL per una variabile stringa significa la deinizializzazione completa di questa variabile.

La costante WRONG_VALUE è destinata per i casi, dove è necessario restituire un valore di un enumerazione, e questo deve essere un valore errato. Per esempio, quando abbiamo bisogno di

informare che il valore di ritorno è un valore da questa enumerazione. Consideriamo a titolo di esempio alcune funzioni CheckLineStyle function(), che restituiscono lo stile di linea di un oggetto, specificato dal suo nome. Se al momento del check stile ObjectGetInteger() il risultato è vero, viene restituito un valore compreso tra ENUM_LINE_STYLE, altrimenti viene restituito WRONG_VALUE.

```
voidOnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Errore di ottenimento stile linea.");
  }
//+------------------------------------------------------------------------------+
//| restituisce lo stile della linea per un oggetto specificato dal suo nome     |
//+------------------------------------------------------------------------------+
ENUM_LINE_STYLE CheckLineStyle(string name)
  {
   long style;
//---
   if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
      return((ENUM_LINE_STYLE)style);
   else
      return(WRONG_VALUE);
  }
```

La costante WHOLE_ARRAY è destinata a funzioni che richiedono di specificare il numero di elementi di array trasformati:

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

Se si desidera specificare, che tutti i valori della matrice da una posizione specificata fino alla fine devono essere elaborati, è necessario specificare solo il valore WHOLE_ARRAY.

IS_PROFILE_MODE (costante) permette di cambiare un'operazione di programma per la raccolta di dati corretti in modalità profiling. Profiling consente di misurare il tempo di esecuzione dei singoli frammenti di programma (di solito comprende le funzioni), nonché il calcolo del numero di tali chiamate. La chiamata della funzione Sleep() può essere disattivata per determinare il tempo di esecuzione in modalità di profiling, come in questo esempio:

```
//--- Sleep può effettivamente influenzare (cambiare) il risultato del profiling
if(!IS_PROFILE_MODE) Sleep(100); // disabilita la chiamata Sleep() in modalità profil
```

Il valore costate IS_PROFILE_MODE è impostato dal compilatore durante la compilazione, mentre è impostato a zero in modo convenzionale. Quando si lancia un programma in modalità profiling, viene eseguita una raccolta speciale e IS_PROFILE_MODE viene sostituito con un valore diverso da zero.

La costante IS_DEBUG_MODE può essere utile quando è necessario modificare un po' il funzionamento di un programma MQL5 in modalità di debug. Ad esempio, in modalità di debug può essere necessario visualizzare ulteriori informazioni di debug nel registro terminale o creare altri oggetti grafici in un grafico.

L'esempio seguente crea un oggetto Label e imposta la sua descrizione e il colore a seconda della modalità script in esecuzione. Per eseguire uno script in modalità di debug da MetaEditor, premere F5. Se si esegue lo script dalla finestra del browser nel terminale, il colore e il testo dell' oggetto Label sarà diverso.

Esempio:

```
//+------------------------------------------------------------------------------------+
//|                                                  Check_DEBUG_MODE.mq5 |
//|                        Copyright © 2009, MetaQuotes Software Corp. //| |
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object",label_name,"not found. Error code = ",GetLastError());
      //--- crea Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- imposta coordinate X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- imposta coordinate Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // modalità debug
        {
         //--- mostra il messaggio riguardo la modalità di esecuzione dello script
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- imposta il colore del testo in rosso
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("Impossibile impostare il colore. Error",GetLastError());
        }
      else               // modalità operazione
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- imposta il colore del testo ad invisibile
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Impossibile impostare il colore. Error ",GetLastError());
```

```
         }
      ChartRedraw();
      DebugBreak();     // qui avverrà la terminazione, se siamo in modalità debug
     }
  }
```

Crypt Methods

The ENUM_CRYPT_METHOD enumeration is used to specify the data tranformation method, used in CryptEncode() and CryptDecode() functions.

ENUM_CRYPT_METHOD

Constant	Description
CRYPT_BASE64	BASE64
CRYPT_AES128	AES encryption with 128 bit key (16 bytes)
CRYPT_AES256	AES encryption with 256 bit key (32 bytes)
CRYPT_DES	DES encryption with 56 bit key (7 bytes)
CRYPT_HASH_SHA1	SHA1 HASH caculation
CRYPT_HASH_SHA256	SHA256 HASH caculation
CRYPT_HASH_MD5	MD5 HASH caculation
CRYPT_ARCH_ZIP	ZIP archives

Vedi anche

DebugBreak, Proprietà di esecuzione programma MQL5, CryptEncode(), CryptDecode()

Strutture Dati

Il linguaggio MQL5 offre 12 strutture predefinite:

- MqlDateTime è designata per lavorare con data ed ora;
- MqlParam può inviare parametri di input quando si crea un handle dell'indicatore utilizzando la funzione IndicatorCreate();
- MqlRates è designata per manipolare i dati storici, essa contiene informazioni riguardanti il prezzo, il volume e lo spread;
- MqlBookInfo è designata per ottenere informazioni sul Depth of Market;
- MqlTradeRequest viene utilizzata per la creazione di una richiesta di trade per le operazioni di trade;
- MqlTradeCheckResult viene usata per controllare la preparazione della richiesta di trade prima dell'invio di questa;
- MqlTradeResult contiene una risposta del trade server ad una richiesta di trade, inviata dalla funzione OrderSend();
- MqlTradeTransaction contiene la descrizione della transazione di trade;
- MqlTick è designata per il recupero rapido delle informazioni più richieste sui prezzi attuali.
- Le strutture del Calendario Economico sono utilizzate per ottenere dati sugli eventi del calendario economico inviati alla piattaforma MetaTrader 5 in tempo reale. Le funzioni del calendario economico permettono di analizzare i parametri macroeconomici immediatamente dopo il rilascio di nuovi report, poiché i valori rilevanti vengono trasmessi direttamente dalla sorgente senza alcun ritardo.

MqlDateTime

La struttura tipo di data contiene otto campi di tipo int:

```
struct MqlDateTime
  {
   int year;            // Anno
   int mon;             // Mese
   int day;             // Giorno
   int hour;            // Ora
   int min;             // Minuti
   int sec;             // Secondi
   int day_of_week;     // Giorni della settimana (0-Domenica, 1-Lunedì, ... ,6-Sabato)
   int day_of_year;     // Numedo di giorno dell'anno (al 1mo Gennaio viene assegnato
  };
```

Nota

Il numero del giorno dell'anno day_of_year per l'anno bisestile, da Marzo, sarà diverso da un numero del giorno corrispondente per un anno non bisestile.

Esempio:

```
voidOnStart()
  {
//---
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, giorni dell'anno = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  Risultato:
   01.03.2008, day of year = 60
   01.03.2009, day of year = 59
*/
```

Vedi anche

TimeToStruct, Strutture e Classi

La struttura dei parametri di input degli Indicatori (MqlParam)

La struttura MqlParam è stata appositamente progettata per fornire parametri di input quando si crea l'handle di un indicatore tecnico utilizzando la funzione IndicatorCreate ().

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                  // tipo di parametro di input, valore di
   long              integer_value;         // campo per memorizzare un tipo intege
   double            double_value;          // campo per memorizzare il tipo double
   string            string_value;          // campo per memorizzare il tipo string
  };
```

Tutti i parametri di input di un indicatore vengono trasmessi sottoforma di array di tipo MqlParam, il campo *type* di ciascun elemento di questa matrice specifica il tipo di dati trasmessi dall'elemento. I valori degli indicatori devono essere prima immessi nei campi appropriati per ogni elemento (in *integer_value*, In *double_value* o *valore_stringa*) a seconda del valore dell' ENUM_DATATYPE enumerazione che è specificata nel campo *type*(campo).

Se il valore IND_CUSTOM viene passato per terzo come tipo di indicatore alla funzione IndicatorCreate(), il primo elemento dell'array di parametri di input deve avere il campo *type* con il valore di TYPE_STRING dall'enumerazione ENUM_DATATYPE, ed il campo *string_value* deve contenere il nome dell' indicatore personalizzato.

MqlRates

Questa struttura memorizza le informazioni sui prezzi, i volumi e lo spread.

```
struct MqlRates
  {
   datetime time;         // Orario di inizio del periodo
   double   open;         // Prezzo di apertura
   double   high;         // Il prezzo più alto del periodo
   double   low;          // Il prezzo più basso del periodo
   double   close;        // Prezzo close
   long     tick_volume;  // Volume Tick
   int      spread;       // Spread
   long     real_volume;  // Volume Trade
  };
```

Esempio:

```
voidOnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("Errore nella copia dei dati dei prezzi ",GetLastError());
   else Print("Copiate ",ArraySize(rates)," barre");
  }
```

Vedi anche

CopyRates, Accesso al TimeSeries

MqlBookInfo

Fornisce informazioni sui dati di profondità del mercato.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    type;           // Tipo di ordine dall'enumerazione ENUM_BOOK_TYP
   double            price;          // Prezzo
   long              volume;         // Volume
   double            volume_real;    // Volume con maggiore accuratezza
  };
```

Nota

La struttura MqlBookInfo è predefinita, pertanto non richiede alcuna dichiarazione e descrizione. Per utilizzare la struttura, basta dichiarare una variabile di questo tipo.

Il DOM è disponibile solo per alcuni simboli.

Esempio:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo su ",Symbol());
     }
   else
     {
      Print("Fallimento nel ricevere il DOM per il simbolo",Symbol());
     }
```

Vedi anche

MarketBookAdd, MarketBookRelease, MarketBookGet, Ordini di Trade nel DOM, Tipi di Dati

La struttura Trade Request (MqlTradeRequest)

L'interazione tra il terminale client ed il trade server per l'esecuzione dell'operazione di piazzamento dell'ordine viene eseguita utilizzando richieste di trade. La richiesta di trade è rappresentata dalla predefinita speciale struttura di tipo MqlTradeRequest, che contiene tutti i campi necessari per eseguire affari di trade. Il risultato della richiesta di elaborazione è rappresentato dalla struttura di tipo MqlTradeResult.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;           // Tipo d'operazione di trade
   ulong                         magic;            // Expert Advisor ID (magic number)
   ulong                         order;            // Ticket dell'ordine
   string                        symbol;           // Simbolo di trade
   double                        volume;           // Volume richiesto per un affare,
   double                        price;            // Prezzo
   double                        stoplimit;        // Livello StopLimit dell'ordine
   double                        sl;               // Livello Stop Loss dell'ordine
   double                        tp;               // Take Profit level dell'ordine
   ulong                         deviation;        // Deviazione massima possibile da
   ENUM_ORDER_TYPE               type;             // Tipo di ordine
   ENUM_ORDER_TYPE_FILLING       type_filling;     // Tipo d'esecuzione dell'ordine
   ENUM_ORDER_TYPE_TIME          type_time;        // Tipo d'espirazione dell'ordine
   datetime                      expiration;       // Orario d'espirazione dell'ordine
   string                        comment;          // Commento dell'ordine
   ulong                         position;         // Ticket della posizione
   ulong                         position_by;      // Il ticket di una posizione oppos
  };
```

Descrizione campi

Campi	Descrizione
action	Tipo d'operazione di trade. Può essere uno dei valori dell'enumerazione ENUM_TRADE_REQUEST_ACTIONS.
magic	Expert Advisor ID. Esso permette di organizzare l'elaborazione analitica degli ordini di trade. Ogni Expert Advisor può impostare il proprio ID univoco per l'invio di una richiesta di trade.
order	Ticket dell' ordine. E' utilizzato per modificare ordini pendenti.
symbol	Simbolo dell'ordine. Non è necessario per la modifica dell'ordine ed operazioni di chiusura di posizione.
volume	Volume richiesto dell'ordine in lotti. Si noti che il volume reale di un affare dipenderà dal tipo d'esecuzione dell'ordine.
price	Prezzo, raggiunto il quale l'ordine deve essere eseguito. Ordini di mercato di simboli, il cui tipo di esecuzione è "Esecuzione a

Campi	Descrizione
	mercato" (SYMBOL_TRADE_EXECUTION_MARKET), Di TRADE_ACTION_DEAL Digitare, Non richiedono specifiche del prezzo.
stoplimit	Il valore del prezzo, al quale l'ordine pendente Limit sarà piazzato, quando il prezzo raggiunge il *prezzo*(valore) (questa condizione è obbligatoria). Fino ad allora l'ordine pendente non viene piazzato.
sl	Prezzo Stop Loss in caso di movimento di prezzo sfavorevole
tp	Prezzo Take Profit in caso di movimento di prezzo favorevole
deviation	La deviazione prezzo massima, specificata in punti
type	Tipo ordine. Può essere uno dei valori dell'enumerazione ENUM_ORDER_TYPE .
type_filling	Tipo d'esecuzione dell'ordine. Può essere uno dei valori dell'enumerazione ENUM_ORDER_TYPE_FILLING.
type_time	Tipo di scadenza dell'ordine. Può essere uno dei valori dell'enumerazione ENUM_ORDER_TYPE_TIME.
expiration	Oradio di espirazione dell'ordine (per ordini di tipo ORDER_TIME_SPECIFIED)
comment	Commento dell'ordine
position	Ticket di una posizione. Dovrebbe essere riempito quando una posizione viene modificata o chiusa per identificare la posizione. Di regola è uguale al ticket dell'ordine, sulla base del quale è stata aperta la posizione.
position_by	Ticket di una posizione opposta. Utilizzata quando una posizione viene chiusa da una contrapposta aperta per lo stesso simbolo nella direzione opposta.

Quando si modifica o si chiude una posizione nel sistema di copertura (hedging), assicurarsi di specificare il suo ticket (MqlTradeRequest::ticket). Il ticket può anche essere specificato nel sistema di compensazione(netting), anche se una posizione è identificata dal nome del simbolo.

Per l'invio di ordini per eseguire operazioni di trade è necessario utilizzare la funzione OrderSend(). Per ogni operazione di trade è necessario specificare i campi obbligatori; campi facoltativi possono anche essere riempiti. Ci sono sette casi possibili per inviare un ordine di trade:

Esecuzione a richiesta

Si tratta di un ordine di trade per aprire una posizione in modalità di Esecuzione della Richiesta (trade sui prezzi richiesti). Si richiede di specificare i seguenti 9 campi:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type

- type_filling

Inoltre è possibile specificare i valori dei campi "magic" e "field".

Esecuzione immediata

Si tratta di un ordine di trade per aprire una posizione in modalità esecuzione immediata (trade ai prezzi correnti). Richiede specificazione dei seguenti 9 campi:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

Inoltre è possibile specificare i valori dei campi "magic" e "field".

Esecuzione a mercato

Si tratta di un ordine di trade per aprire una posizione in modalità di esecuzione a mercato. Si richiede di specificare i seguenti 5 campi:

- action
- symbol
- volume
- type
- type_filling

Inoltre è possibile specificare i valori dei campi "magic" e "field".

Esecuzione a cambio

Si tratta di un ordine di trade per aprire una posizione in modalità di esecuzione Exchange. Si richiede di specificare i seguenti 5 campi:

- action
- symbol
- volume
- type
- type_filling

Inoltre è possibile specificare i valori dei campi "magic" e "field".

Esempio per TRADE_ACTION_DEAL, operazione di trade per aprire una posizione Buy:

```
#define EXPERT_MAGIC 123456   // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Apre posizione Buy                                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parametri della richiesta
   request.action   =TRADE_ACTION_DEAL;                     // tipo di operazione di t
   request.symbol   =Symbol();                              // simbolo
   request.volume   =0.1;                                   // volume di 0.1 lotti
   request.type     =ORDER_TYPE_BUY;                        // tipo di ordine
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // prezzo per l'apertura
   request.deviation=5;                                     // permessa la deviazione
   request.magic    =EXPERT_MAGIC;                          // MagicNumber dell'ordine
//--- invia la richiesta
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se impossibilitato ad i
//--- informazione riguardo l'operazione
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Esempio di TRADE_ACTION_DEAL, operazione di trade per aprire la posizione Sell:

```
#define EXPERT_MAGIC 123456   // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Apre posizione Sell                                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parametri della richiesta
   request.action   =TRADE_ACTION_DEAL;                     // tipo di operazione di t
   request.symbol   =Symbol();                              // simbolo
   request.volume   =0.2;                                   // volume di 0.2 lotti
   request.type     =ORDER_TYPE_SELL;                        // tipo di ordine
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // prezzo per l'apertura
   request.deviation=5;                                     // permessa la deviazione
   request.magic    =EXPERT_MAGIC;                          // MagicNumber dell'ordine
//--- invia la richiesta
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se impossibilitato ad i
//--- informazione riguardo l'operazione
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Esempio di TRADE_ACTION_DEAL, operazione di trade per chiudere la posizione:

```
#define EXPERT_MAGIC 123456   // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Chiude tutte le posizioni                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // numero della posizione aperta
//--- itera su tutte le posizioni
   for(int i=total-1; i>=0; i--)
     {
      //--- parametri per l'ordine
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- informazioni output riguardo la posizione
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- se il MagicNumber corrisponde
      if(magic==EXPERT_MAGIC)
        {
         //--- azzera la richiesta ed i valori risultato
         ZeroMemory(request);
         ZeroMemory(result);
         //--- imposta i parametri dell'operazione
         request.action   =TRADE_ACTION_DEAL;        // tipo di operazione di trade
         request.position =position_ticket;          // ticket della posizione
         request.symbol   =position_symbol;          // simbolo
         request.volume   =volume;                   // volume della posizione
         request.deviation=5;                        // deviazione consentita dal prez
         request.magic    =EXPERT_MAGIC;             // MagicNumber della posizione
         //--- imposta il prezzo ed il tipo d'ordine a seconda del tipo della posizion
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- informazioni output riguardo la chiusura
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToStrinq
         //--- imposta la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend errore %d",GetLastError());  // se impossibile invi
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

```
      }
   }
//+------------------------------------------------------------------+
```

SL & TP Modifica

Trade order per modificare i prezzi di StopLoss e/o i prezzi TakeProfit. Si richiede di specificare i seguenti 4 settori:

- action
- symbol
- sl
- tp
- position

Esempio per TRADE_ACTION_SLTP, operazione di trade per la modifica dei valori di Stop Loss e Take Profit di una posizione aperta:

```
#define EXPERT_MAGIC 123456  // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Modifica dello Stop Loss e Take Profit della posizione           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // numero delle posizioni aperte
//--- itera su tutte le posizioni/t34>
   for(int i=0; i<total; i++)
     {
      //--- parametri dell'ordine
      ulong  position_ticket=PositionGetTicket(i);// ticket della posizione
      string position_symbol=PositionGetString(POSITION_SYMBOL); // simbolo
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // numero d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber della posizione
      double volume=PositionGetDouble(POSITION_VOLUME);    // volume della posizione
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss della posizione
      double tp=PositionGetDouble(POSITION_TP);  // Take Profit della posizione
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- output informazioni riguardo la posizione
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se il MagicNumber corrisponde, Stop Loss e Take Profit non sono stati defi
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

```
         //--- calcola i livelli del prezzo corrente
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_L
         double price_level;
         //--- se l'offset distanza minima consentita in punti dal corrente punto di c
         if(stop_level<=0)
            stop_level=150; // imposta l'offset distanza di 150 punti dal corrente pun
         else
            stop_level+=50; // imposta l'offset distanza a (SYMBOL_TRADE_STOPS_LEVEL +

         //--- calcolo ed arrotondamento dei valori Stop Loss e Take Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(bid+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(ask-price_level,digits);
           }
         //--- azzero la richiesta ed i risultanti valori
         ZeroMemory(request);
         ZeroMemory(result);
         //--- setting the operation parameters
         request.action  =TRADE_ACTION_SLTP; // type of trade operation
         request.position=position_ticket;   // ticket of the position
         request.symbol=position_symbol;     // simbolo
         request.sl      =sl;                // Stop Loss della posizione
         request.tp      =tp;                // Take Profit della posizione
         request.magic=EXPERT_MAGIC;         // MagicNumber della posizione
         //--- output informazioni riguardo la modifica
         PrintFormat("Modifica #%I64d %s %s",position_ticket,position_symbol,EnumToStr
         //--- invia la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // se impossibile invia
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Ordine Pendente

Ordine di trade per piazzare un ordine pendente. Si richiede di specificare i seguenti 11 campi:

- action
- symbol
- volume
- price
- stoplimit
- sl
- tp
- type

- type_filling
- type_time
- expiration

Inoltre è possibile specificare i valori dei campi "magic" e "field".

Esempio di TRADE_ACTION_PENDING, operazione di trade per piazzare un ordine pendente:

```
#property description "Esempio di piazzamento dell'ordine pendente"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // MagicNumber dell'expert
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;    // tipo d'ordine
//+------------------------------------------------------------------+
//| Piazza l'ordine pendente                                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parametri per piazzare un ordine pendente
   request.action   =TRADE_ACTION_PENDING;                          // tipo di oper
   request.symbol   =Symbol();                                      // simbolo
   request.volume   =0.1;                                           // volume di 0.
   request.deviation=2;                                             // ammonetare c
   request.magic    =EXPERT_MAGIC;                                  // MagicNumber
   int offset = 50;                                                 // offset from
   double price;                                                    // prezzo d'inn
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);             // valore del p
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);             // numero di ci
   //--- controllo del tipo d'operazione
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                      // tipo d'ordin
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // prezzo per l
      request.price    =NormalizeDouble(price,digits);             // prezzo d'ape
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                      // tipo d'ordi
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;     // prezzo per
      request.price    =NormalizeDouble(price,digits);              // prezzo d'ap
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                           // tipo d'ordi
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // prezzo per
      request.price=NormalizeDouble(price,digits);                 // prezzo d'ap
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                       // tipo d'ordi
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;     // prezzo per
      request.price    =NormalizeDouble(price,digits);              // prezzo d'ap
     }
   else Alert("Questo esempio è solo per piazzare ordini pendenti");   // se non ci so
//--- invia la richiesta
   if(!OrderSend(request,result))
      PrintFormat("OrderSend errore %d",GetLastError());             // se impossi
//--- informazioni riguardo l'operazione
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Modificare Ordine Pendente

Ordine di trade per modificare i prezzi di un ordine pendente. Si richiede di specificare i seguenti 7 campi:

- action
- order
- price
- sl
- tp
- type_time
- expiration

Esempio di TRADE_ACTION_MODIFY, operazione di trade per la modifica dei livelli di prezzo di ordini pendenti:

```
#define EXPERT_MAGIC 123456  // MagicNumber dell'expert
//+------------------------------------------------------------------+
//| Modifica degli ordini pendenti                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // numero totale di orini pendenti piazzati
//--- itera su tutti gli ordini pendenti piazzati
   for(int i=0; i<total; i++)
     {
      //--- parametri dell'ordine
      ulong  order_ticket=OrderGetTicket(i);                          // order tick
      string order_symbol=Symbol();                                   // simbolo
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // numero di
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // MagicNumbe
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // corrente v
      double sl=OrderGetDouble(ORDER_SL);                             // corrente S
      double tp=OrderGetDouble(ORDER_TP);                             // corrente T
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // tipo di or
      int offset = 50;                                                // offset dal
      double price;                                                   // prezzo d'i
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // valore in
      //--- output informazioni riguardo l'ordine
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se il MagicNumber corrisponde, Stop Loss e Take Profit non sono definiti
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                          // tipo d'opera
         request.order = OrderGetTicket(i);                           // ticket ordi
         request.symbol   =Symbol();                                  // simbolo
         request.deviation=5;                                         // deviazione
        //--- imposta il livello del prezzo, Take Profit e Stop Loss dell'ordine diper
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prezzo
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                // prezzo
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // prezzo
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // prezzo
           }
         //--- invia la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend errore %d",GetLastError());  // se impossibile invi
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- azzera la richiesta ed i risultanti valori
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Eliminare Ordine Pendente

Ordine di trade per eliminare un ordine pendente. Richiede di specificare i 2 campi seguenti:

- action
- order

Esempio per TRADE_ACTION_REMOVE, operazione di trade per eliminare ordini pendenti:

```
#define EXPERT_MAGIC 123456  // MagicNumber dell'expert
```

```
//+------------------------------------------------------------------+
//| Elimina ordini pendenti                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara ed inizializza la richiesta di trade ed il risultato della richiesta di
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // numero totale di ordini pendenti piazzati
//--- itera su tutti gli ordini pendenti piazzati
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // ticket ordine
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // MagicNumber dell'ord
      //--- se il MagicNumber corrisponde
      if(magic==EXPERT_MAGIC)
        {
         //--- azzera la richiesta ed i valori del risultato
         ZeroMemory(request);
         ZeroMemory(result);
         //--- imposta i parametri dell'operazione
         request.action=TRADE_ACTION_REMOVE;                   // tipo d'operazione di
         request.order = order_ticket;                         // ticket ordine
         //--- invia la richiesta
         if(!OrderSend(request,result))
            PrintFormat("OrderSend errore %d",GetLastError());  // se impossibile invi
         //--- informazioni riguardo l'operazione
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Guarda anche

Strutture e Classi, Funzioni di Trade, Proprietà Ordini

La struttura dei Risultati di un Controllo di Richiesta di Trade (MqlTradeCheckResult)

Prima dell'invio ad una richiesta per un' operazione di trade ad un trade server, si consiglia di controllarla. Il controllo viene eseguito utilizzando la funzione OrderCheck(), per cui vengono passati la richiesta controllata ed una variabile di tipo struttura MqlTradeCheckResult. Il risultato del controllo verrà scritto in questa variabile.

```
struct MqlTradeCheckResult
  {
   uint          retcode;             // Codice di risposta
   double        balance;             // Bilancio dopo l'esecuzione di un deal
   double        equity;              // Equità dopo l'esecuzione di un deal
   double        profit;              // Profitto fluttuante
   double        margin;              // Requisiti di margine
   double        margin_free;         // Margine libero
   double        margin_level;        // Livello del margine
   string        comment;             // Commento al codice di risposta (descrizione de
  };
```

Descrizione dei campi

Campo	Descrizione
retcode	Il codice di ritorno
balance	Valore del Bilancio che risulterà dopo l'esecuzione dell'operazione di trade
equity	Valore dell' Equitò che risulterà dopo l'esecuzione dell'operazione di trade
profit	Valore del profitto fluttuante che risulterà dopo l'esecuzione dell'operazione di trade
margin	Margine richiesto per l'operazione di trade
margin_free	Margine libero che varrà lasciato dopo l'esecuzione dell'operazione di trade
margin_level	Livello del Margine che verrà fissato dopo l'esecuzione dell'operazione di trade
riga	Commento al codice di risposta, descrizione di errore

Vedi anche

Trade Request Structure, Struttura per i Prezzi Correnti, OrderSend, OrderCheck

La struttura di un risultato di Richiesta di Trade (MqlTradeResult)

Come risultato di una richiesta di trade, un trade server restituisce i dati relativi al risultato dell'elaborazione della richiesta di trade come una speciale struttura predefinita di tipo MqlTradeResult.

```
struct MqlTradeResult
  {
   uint     retcode;          // Codice di ritorno dell'operazione
   ulong    deal;             // Ticket dell'Affare, se è stato eseguito
   ulong    order;            // Ticket dell'Ordine, se è stato piazzato
   double   volume;           // Volume dell'Affare, confermato dal broker
   double   price;            // Prezzo dell'affare, confermato dal broker
   double   bid;              // Prezzo Bid Corrente
   double   ask;              // Prezzo Ask Corrente
   string   comment;          // Commento del Broker all'operazione (per impostazione
   int      request_id;       // ID della Richiesta impostato dal terminale durante l
  };
```

Descrizione dei Campi

Campo	Descrizione
retcode	Il codice di ritorno del trade server
deal	Ticket Affare, se un affare è stata eseguito. E' disponibile per un'operazione di trade di tipo TRADE_ACTION_DEAL
ordine	Ticket Ordine, se il ticket è stato piazzato. E' disponibile per un'operazione di trade di tipo TRADE_ACTION_PENDING
volume	Volume Deal, confermato dal broker. Dipende dal tipo di riempimento dell'ordine
price	Prezzo Deal, confermato dal broker. Dipende dal campo *deviazione* della richiesta di trade e/o sull'operazione di trade
bid	L'attuale prezzo di mercato Bid (prezzo riquotazione)
ask	Il corrente prezzo Ask di mercato (prezzo riquotazione)
riga	Il commento del broker all'operazione (per impostazione predefinita è riempito dalla descrizione Il codice di ritorno del trade server)
request_id	ID richiesta impostato dal terminale per l'invio trade server

Il risultato dell'operazione di trade viene restituito ad una variabile di tipo MqlTradeResult, che viene passata come secondo parametro ad OrderSend() per effettuare operazioni di trade.

Il terminale fissa la richiesta ID nel campo request_id quando si invia al trade server usando le funzioni OrdersSend() e OrderSendAsync(). Il terminale riceve i messaggi sulle operazioni effettuate dal trade server e li sottopone per l'elaborazione della funzione OnTradeTransaction() contenente i seguenti componenti come parametri:

- descrizione dell'operazione di trade nella struttura MqlTradeTransaction;
- descrizione della richiesta di trade inviata dalla funzione OrderSend() o OrdersSendAsync(). L' ID richiesta viene inviato dal terminale al trade server, mentre la stessa richiesta e la sua request_id sono memorizzate nella memoria del terminale;
- il risultato della richiesta di esecuzione come nella struttura MqlTradeResult con il campo request_id contenente l'ID della richiesta.

La funzione OnTradeTransaction() riceve tre parametri di input, ma gli ultimi due devono essere analizzati solo per le operazioni aventi tipo TRADE_TRANSACTION_REQUEST. In tutti gli altri casi, i dati sulla richiesta di trade ed il suo risultato dell'esecuzione non vengono riempiti. Esempio di analisi dei parametri può essere trovato nella Struttura di una Richiesta di Trade.

L'impostazione di request_id dal terminale per la richiesta di trade, quando si al trade server è principalmente introdotta per lavorare con la funzione asincrona OrderSendAsync(). Questo identificatore consente di associare l'azione eseguita (chiamate di funzioni OrderSend o OrderSendAsync) con il risultato di questa azione inviato ad OnTradeTransaction().

Esempio:

```
//+------------------------------------------------------------------------------+
//| Invia una richiesta di trade con il risultato dell'esecuzione                |
//+------------------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- resetta l'ultimo codice dell'errore, a zero
   ResetLastError();
//--- invia richiesta
   bool success=OrderSend(request,result);
//--- se il risultato fallisce - prova a cercare il perchè
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog: Richiesta di trade fallita. Error = ",GetLastError());
      switch(answer)
        {
         //--- requote
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- l'ordine non è accettato dal server
         case 10006:
           {
            Print("TRADE_RETCODE_REJECT");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
```

```
            }
          //--- prezzo non valido
          case 10015:
            {
             Print("TRADE_RETCODE_INVALID_PRICE");
             Print("request.price = ",request.price,"   result.ask = ",
                   result.ask," result.bid = ",result.bid);
             break;
            }
          //--- SL e/o TP non validi
          case 10016:
            {
             Print("TRADE_RETCODE_INVALID_STOPS");
             Print("request.sl = ",request.sl," request.tp = ",request.tp);
             Print("result.ask = ",result.ask," result.bid = ",result.bid);
             break;
            }
          //--- volume non valido
          case 10014:
            {
             Print("TRADE_RETCODE_INVALID_VOLUME");
             Print("request.volume = ",request.volume,"   result.volume = ",
                   result.volume);
             break;
            }
          //--- non ci sono soldi a sufficienza per l'operazione di trade
          case 10019:
            {
             Print("TRADE_RETCODE_NO_MONEY");
             Print("request.volume = ",request.volume,"   result.volume = ",
                   result.volume,"   result.comment = ",result.comment);
             break;
            }
          //--- alcune altre ragioni, in output il codice di risposta del server
          default:
            {
             Print("Altra risposta = ",answer);
            }
         }
       //--- notifica circa il risultato non riuscito della richiesta di trade, restitu
       return(false);
      }
//--- OrderSend() restituisce true - ripete la risposta
   return(true);
  }
```

Struttura di una Transazione di Trade (MqlTradeTransaction)

Quando si eseguono alcune azioni precise su un trade account, il suo stato cambia. Tali azioni comprendono:

- Inviare una richiesta di trade da qualsiasi applicazione MQL5 nel terminale client utilizzando le funzioni OrderSend e OrderSendAsync e la sua ulteriore esecuzione;
- Inviare una richiesta di trade tramite l'interfaccia grafica del terminale e la sua esecuzione ulteriore;
- Attivazione di ordini pendenti ed ordini di stop sul server;
- Esecuzione di operazioni sul lato trade server.

Le operazioni commerciali di seguito vengono eseguite come risultato di queste azioni:

- gestione di una richiesta di trade;
- cambio di ordini aperti;
- cambio della cronistoria degli ordini;
- cambio della cronistoria degli affari;
- cambio delle posizioni.

Per esempio, quando si invia un ordine di acquisto di mercato, esso viene gestito, un ordine di acquisto appropriata viene creato per l'account, l'ordine quindi viene eseguito e rimosso dalla lista di quelli aperti, e poi viene aggiunto alla cronistoria ordini, un appropriato aff viene aggiunto alla cronistoria ed una nuova posizione viene creata. Tutte queste azioni sono transazioni di trade.

Lo Speciale handler OnTradeTransaction() viene fornito in MQL5 per ottenere transazioni di trade applicate ad un account. Il primo parametro dell'handler ottiene la struttura MqlTradeTransaction descrivendo transazioni di trade.

```
struct MqlTradeTransaction
  {
   ulong                     deal;             // Ticket dell'Affare
   ulong                     order;            // Ticket dell'Ordine
   string                    symbol;           // Nome del simbolo di Trade
   ENUM_TRADE_TRANSACTION_TYPE type;           // Tipo del simbolo di Trade
   ENUM_ORDER_TYPE           order_type;       // Tipo di Ordine
   ENUM_ORDER_STATE          order_state;      // Stato dell'Ordine
   ENUM_DEAL_TYPE            deal_type;        // Tipo dell'Affare
   ENUM_ORDER_TYPE_TIME      time_type;        // Tipo di Ordine per periodo dell
   datetime                  time_expiration;  // Orario di espirazione dell'Ordi
   double                    price;            // Prezzo
   double                    price_trigger;    // Prezzo di attivazione dell'ordi
   double                    price_sl;         // Livello Stop Loss
   double                    price_tp;         // Livello Take Profit
   double                    volume;           // Volume in lotti
   ulong                     position;         // Position ticket
   ulong                     position_by;      // Ticket of an opposite position
  };
```

Descrizione Campi

Campo	Descrizione
deal	Ticket affare.
ordine	Ticket dell' Ordine.
symbol	Il nome del simbolo di trade, per il quale viene eseguita l'operazione.
type	Tipo di transazione di trade. Il valore può essere uno dei valori dell'enumerazione ENUM_TRADE_TRANSACTION_TYPE.
order_type	Tipo ordine di trade. Il valore può essere uno dei valori dell'enumerazione ENUM_ORDER_TYPE.
order_state	Stato dell'ordine di trade . Il valore può essere uno dei valori dell'enumerazione ENUM_ORDER_STATE.
deal_type	Tipo di affare. Il valore può essere uno dei valori dell'enumerazione ENUM_DEAL_TYPE.
time_type	Tipo di ordine all'espirazione. Il valore può essere uno dei valori ENUM_ORDER_TYPE_TIME.
time_expiration	Termine di espirazione dell'ordine (per ordini dei tipi ORDER_TIME_SPECIFIED e ORDER_TIME_SPECIFIED_DAY).
price	Prezzo. A seconda del tipo di transazione di trade, può essere il prezzo di un ordine, un affare o una posizione.
price_trigger	Prezzo di stop(attivazione) dell'ordine di stop limit (ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT).
price_sl	Prezzo di Stop Loss. A seconda del tipo di transazione di trade, può riferirsi ad un ordine, un affare o una posizione.
price_tp	Prezzo Take Profit. A seconda del tipo di transazione di trade, può riferirsi ad un ordine, un affare o una posizione.
volume	Volume in lotti. A seconda del tipo di transazione di trade, potrebbe indicare il volume attuale di un ordine, di un affare o di una posizione.

Il parametro essenziale per l'analisi della transazione ricevuta è il tipo specificato nel campo **type**. Ad esempio, se una transazione è di tipo TRADE_TRANSACTION_REQUEST (viene ricevuto il risultato dell' handling di una richiesta di trade dal server), la struttura ha solo un campo che viene riempito completamente - **type**. Altri campi non vengono analizzati. In questo caso, si possono analizzare due ulteriori **richieste** ed i parametri **risultato** presentati all'handler OnTradeTransaction(), come mostrato di seguito.

Avendo i dati su un tipo di operazione di trading, si può decidere l'analisi dello stato attuale di ordini, le posizioni e le offerte per un conto di trading. Ricordate che una richiesta di trade inviata al server dal terminale può generare generare diverse nuove operazioni. La priorità del loro arrivo presso il terminale non è garantita.

La struttura MqlTradeTransaction viene riempita in modi diversi a seconda del tipo di transazione di trade (ENUM_TRADE_TRANSACTION_TYPE):

TRADE_TRANSACTION_ORDER_* and TRADE_TRANSACTION_HISTORY_*

I seguenti campi nella struttura MqlTradeTransaction sono riempiti per le transazioni di trade relative alla gestione degli ordini aperti (TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE e TRADE_TRANSACTION_ORDER_DELETE) e la cronistoria degli ordini (TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE):

- order - ticket dell'ordine;
- symbol - nome simbolo dell'ordine;
- type - tipo di transazione di trade;
- order_type - tipo di ordine;
- orders_state - corrente stato dell'ordine;
- time_type - tipo di espirazione dell'ordine;
- time_expiration - orario di espirazione dell'ordine (per ordini aventi i tipi di espirazione ORDER_TIME_SPECIFIED e ORDER_TIME_SPECIFIED_DAY);
- price - order price specified by a client;
- price_trigger - prezzo di stop dell'ordine di stop limit (solo per ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT);
- price_sl - prezzo dell'ordine di Stop Loss (filled, se specificato nell'ordine);
- price_tp - prezzo dell'ordine di Take Profit (filled, se specificato nell'ordine);
- volume - corrente volume dell'ordine (unfilled). Il volume iniziale dell'ordine può essere trovato nella cronistoria degli ordini usando la funzione HistoryOrders *.

TRADE_TRANSACTION_DEAL_*

I seguenti campi nella struttura MqlTradeTransaction vengono riempiti per le transazioni di trade relative all' handling delgli affari (TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE e TRADE_TRANSACTION_DEAL_DELETE):

- deal - ticket dell'affare;
- order - ticket dell'ordine, dove è stato eseguito l'affare su cui si basa;
- symbol - nome del simbolo dell'affare;
- type - tipo di transazione di trade;
- deal_type - tipo di affare;
- price - prezzo dell'affare;
- price_sl - prezzo di Stop Loss (filled, se specificato nell'ordine, dove è stato eseguito l'affare su cui si basa);
- price_tp - prezzo di Take Profit (filled, se specificato nell'ordine, dove è stato eseguito l'affare su cui si basa);
- volume - volume dell'affare in lotti.

TRADE_TRANSACTION_POSITION

I seguenti campi nella struttura MqlTradeTransaction sono riempiti per le transazioni di trade relative alla modifica delle posizioni non connesse all'esecuzione degli affari (TRADE_TRANSACTION_POSITION):

- symbol - nome di posizione del simbolo;
- type - tipo di transazione di trade;
- deal_type - tipo di posizione (DEAL_TYPE_BUY o DEAL_TYPE_SELL);
- prezzo - prezzo di apertura media ponderata della posizione;
- price_sl - prezzo di Stop Loss;

- price_tp - prezzo di Take Profit;
- volume - volume della posizione in lotti, se è stato cambiato.

> Il ambio della posizione (aggiunta, cambio o chiusura), come risultato dell'esecuzione dell'affare, non porta al verificarsi della transazione TRADE_TRANSACTION_POSITION.

TRADE_TRANSACTION_REQUEST

Solo un campo nella struttura MqlTradeTransaction è riempito per le transazioni di trade che descrivono il fatto che una richiesta di trade è stata elaborata dal server ed il risultato dell' elaborazione è stato ricevuto (TRADE_TRANSACTION_REQUEST):

- type - tipo di transazione di trade;

> Solo il campo tipo (tipo di transazione di trade) deve essere analizzato per tali operazioni. Il secondo e terzo parametro della funzione OnTradeTransaction (richiesta e risultato) devono essere analizzati per ulteriori dati.

Esempio:

```
input int MagicNumber=1234567;

//--- abilita la classe di trading CTrade e dichiara la variabile di questa classe
#include <Trade\Trade.mqh>
CTrade trade;
//--- flags per installare ed eliminare gli ordini pendenti
bool pending_done=false;
bool pending_deleted=false;
//--- i ticket degli ordini pendenti verranno memorizzati qui
ulong order_ticket;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- imposta il MagicNumber per machiare tutti i nostri ordini
   trade.SetExpertMagicNumber(MagicNumber);
//--- le richieste di trade verranno eseguite in modalità asincrona usando la funzione
   trade.SetAsyncMode(true);
//--- inizializza la variabile per zero
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                    |
//+------------------------------------------------------------------------------+
void OnTick()
  {
//---installa un ordine pendente
   if(!pending_done)
```

```
     {
      double ask=SymbolInfoDouble(_Symbol,SYMBOL_ASK);
      double buy_stop_price=NormalizeDouble(ask+1000*_Point,(int)SymbolInfoInteger(_Sy
      bool res=trade.BuyStop(0.1,buy_stop_price,_Symbol);
      //--- se la funzione BuyStop() viene eseguita con successo
      if(res)
        {
         pending_done=true;
         //--- ottiene il risultato dell'invio di una richiesta da ctrade
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //--- ottiene request_id per la richiesta inviata
         uint request_id=trade_result.request_id;
         Print("La richiesta è stata inviata per impostare un ordine pendente. Request
         //--- memorizza il ticket dell'ordine (sarà zero se si usa la modalità di inv
         order_ticket=trade_result.order;
         //--- tutto è stato fatto, uscita precoce dall'handler OnTick()
         return;
        }
     }
//--- elimina l'ordine pendente
   if(!pending_deleted)
      //--- controllo addizionale
      if(pending_done && (order_ticket!=0))
        {
         //--- prova ad eliminare l'ordine pendente
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- quando la richiesta di eliminazione viene inviata con successo
         if(res)
           {
            pending_deleted=true;
            //--- ottiene il risultato della richiesta di esecuzione
            MqlTradeResult trade_result;
            trade.Result(trade_result);
            //--- prende l' ID della richiesta dal risultato
            uint request_id=trade_result.request_id;
            //--- visualizza nel Journal
            Print("La richiesta è stata inviata per eliminare un ordine pendente #",or
                  ". Request_ID=",request_id,
                  "\r\n");
            //--- fissa il ticket dell'ordine dal risultato della richiesta
            order_ticket=trade_result.order;
           }
        }
//---
  }
//+------------------------------------------------------------------------------+
//| Funzione TradeTransaction                                                    |
```

```
//+------------------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- ottiene type come valore dell'enumerazione
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- se la transazione è il risultato dell'handling richiesta, verrà visualizzato so
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- visualizza il nome stringa della richiesta maneggiata
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- mostra la descrizione del risultato della richiesta
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- memorizza il ticket dell'ordine per la sua cancellazione al prossimo handl
      if(result.order!=0)
        {
         //--- elimina quest'ordine dal suo ticket alla prossima chiamata OnTick()
         order_ticket=result.order;
         Print(" Ticket dell'ordine pendente ",order_ticket,"\r\n");
        }
     }
   else // mostra la descrizione completa per le transazioni di altro tipo
//--- mostra la descrizione di una transazione ricevuta nel Journal
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------------------+
//| Restituisce la descrizione testuale della transazione                        |
//+------------------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Simbolo: "+trans.symbol+"\r\n";
   desc+="Ticket dell'affare: "+(string)trans.deal+"\r\n";
   desc+="Tipo di Affare: "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Ticket dell'Ordine: "+(string)trans.order+"\r\n";
   desc+="Tipo di Ordine: "+EnumToString(trans.order_type)+"\r\n";
   desc+="Stato dell'Ordine: "+EnumToString(trans.order_state)+"\r\n";
   desc+="Tipo dell'orario dell'Ordine: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Espirazione dell'Ordine: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Prezzo: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Innesco Prezzo: "+StringFormat("%G",trans.price_trigger)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
   desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
```

```
//--- restituisce la stringa ottenuta
   return desc;
  }
//+------------------------------------------------------------------------------+
//| Restituisce la descrizione testuale della transazione                        |
//+------------------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Simbolo: "+request.symbol+"\r\n";
   desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Ticket dell'Ordine: "+(string)request.order+"\r\n";
   desc+="Tipo dell'Ordine: "+EnumToString(request.type)+"\r\n";
   desc+="Riempimento dell'Ordine: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Tipo di orario dell'Ordine: "+EnumToString(request.type_time)+"\r\n";
   desc+="Espirazione dell'Ordine: "+TimeToString(request.expiration)+"\r\n";
   desc+="Prezzo: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Punti di deviazione: "+StringFormat("%G",request.deviation)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
   desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
   desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
   desc+="Commento: "+request.comment+"\r\n";
//--- restituisce la stringa ottenuta
   return desc;
  }
//+------------------------------------------------------------------------------+
//| Restituisce la descrizione testuale del risultato della richiesta di handling |
//+------------------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="ID Richiesta: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Ticket dell'Ordine: "+(string)result.order+"\r\n";
   desc+="Ticket dell'Affare: "+(string)result.deal+"\r\n";
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Prezzo: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Commento: "+result.comment+"\r\n";
//--- restituisce la stringa ottenuta
   return desc;
  }
```

Vedi anche

Tipi di Transazione di Trade, OnTradeTransaction()

La Struttura per il Ritorno dei Prezzi Correnti (MqlTick)

Questa è una struttura per memorizzare gli ultimi prezzi del simbolo. È progettata per il recupero rapido delle informazioni più richieste sui prezzi correnti.

```
struct MqlTick
  {
   datetime     time;          // Orario dell'aggiornamento dell'ultimo prezzo
   double       bid;           // Il corrente prezzo Bid
   double       ask;           // Il corrente prezzo Ask
   double       last;          // Prezzo dell'ultimo affare (Last)
   ulong        volume;        // Volume per il corrente prezzo Last
   long         time_msc;      // Orario dell'ultimo aggiornamento di prezzo in millis
   uint         flags;         // Flag Ticks
   double       volume_real;   // Volume con maggior accuratezza per il corrente press
  };
```

La variabile di tipo MqlTick permette di ottenere valori di Ask, Bid, Last e Volume entro una singola chiamata della funzione SymbolInfoTick().

The parameters of each tick are filled in regardless of whether there are changes compared to the previous tick. Thus, it is possible to find out a correct price for any moment in the past without the need to search for previous values at the tick history. For example, even if only a Bid price changes during a tick arrival, the structure still contains other parameters as well, including the previous Ask price, volume, etc.

You can analyze the tick flags to find out what data have been changed exactly:

- TICK_FLAG_BID - tick has changed a Bid price
- TICK_FLAG_ASK - a tick has changed an Ask price
- TICK_FLAG_LAST - a tick has changed the last deal price
- TICK_FLAG_VOLUME - a tick has changed a volume
- TICK_FLAG_BUY - a tick is a result of a buy deal
- TICK_FLAG_SELL - a tick is a result of a sell deal

Esempio:

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Bid = ",last_tick.bid,
            " Ask = ",last_tick.ask,"  Volume = ",last_tick.volume);
     }
   else Print("SymbolInfoTick() fallito, errore = ",GetLastError());
//---
  }
```

Vedi anche

Strutture e Classi, CopyTicks(), SymbolInfoTick()

Economic Calendar structures

Questa sezione descrive le strutture per lavorare con il calendario economico disponibile direttamente nella piattaforma MetaTrader. Il calendario economico è un'enciclopedia già pronta con descrizioni di indicatori macroeconomici, le loro date di rilascio e gradi di importanza. I valori rilevanti degli indicatori macroeconomici vengono inviati alla piattaforma MetaTrader proprio al momento della pubblicazione e vengono visualizzati su un chart come tag che consente di monitorare visivamente gli indicatori richiesti in base a Paesi, valute e importanza.

Le funzioni del calendario economico consentono di condurre l'analisi automatica degli eventi in arrivo in base a criteri di importanza personalizzati da una prospettiva di coppie di Paesi/valute necessarie.

Le descrizioni dei Paesi sono impostate dalla struttura MqlCalendarCountry. È usata nelle funzioni CalendarCountryById() e CalendarCountries()

```
struct MqlCalendarCountry
  {
   ulong                             id;                // ID del Paese (ISO 31
   string                            name;              // nome del testo del Pa
   string                            code;              // nome codice del Paese
   string                            currency;          // codice valuta del pae
   string                            currency_symbol;   // simbolo di valuta del
   string                            url_name;          // nome del paese utili
  };
```

Le descrizioni degli eventi sono impostate dalla struttura MqlCalendarEvent. È usata nelle funzioni CalendarEventById(), CalendarEventByCountry() e CalendarEventByCurrency()

```
struct MqlCalendarEvent
  {
   ulong                             id;                // ID evento
   ENUM_CALENDAR_EVENT_TYPE          type;              // tipo di evento dall'e
   ENUM_CALENDAR_EVENT_SECTOR        sector;            // settore a cui è colle
   ENUM_CALENDAR_EVENT_FREQUENCY     frequency;         // frequenza degli event
   ENUM_CALENDAR_EVENT_TIMEMODE      time_mode;         // modalità orario event
   ulong                             country_id;        // ID Paese
   ENUM_CALENDAR_EVENT_UNIT          unit;              // unità di misura del v
   ENUM_CALENDAR_EVENT_IMPORTANCE    importance;        // importanza dell'event
   ENUM_CALENDAR_EVENT_MULTIPLIER    multiplier;        // moltiplicatore del va
   uint                              digits;            // numero di posizioni d
   string                            source_url;        // URL di una fonte in d
   string                            event_code;        // codice evento
   string                            name;              // nome del testo dell'e
  };
```

I valori degli eventi sono impostati dalla struttura MqlCalendarValue. È usata nelle funzioni CalendarValueById(), CalendarValueHistoryByEvent(), CalendarValueHistory(), CalendarValueLastByEvent() e CalendarValueLast()

```
struct MqlCalendarValue
  {
   ulong                             id;                // ID valore
   ulong                             event_id;          // ID evento
   datetime                          time;              // data e ora dell'event
   datetime                          period;            // periodo del report de
   int                               revision;          // ha rilasciato la revi
   long                              actual_value;      // valore attuale dell'i
   long                              prev_value;        // valore precedente del
   long                              revised_prev_value; // valore rivisto dell'i
   long                              forecast_value;    // valore di previsione
   ENUM_CALENDAR_EVENT_IMPACT        impact_type;       // impatto potenziale su
  };
```

La frequenza degli eventi è specificata nella struttura MqlCalendarEvent. I valori possibili sono impostati nell'elenco **ENUM_CALENDAR_EVENT_FREQUENCY**

ID	Descrizione
CALENDAR_FREQUENCY_NONE	La frequenza di rilascio non è impostata
CALENDAR_FREQUENCY_WEEK	Rilasciato una volta a settimana
CALENDAR_FREQUENCY_MONTH	Rilasciato una volta al mese
CALENDAR_FREQUENCY_QUARTER	Rilasciato una volta al trimestre
CALENDAR_FREQUENCY_YEAR	Rilasciato una volta all'anno
CALENDAR_FREQUENCY_DAY	Rilasciato una volta al giorno

Il tipo di evento è specificato nella struttura MqlCalendarEvent. I valori possibili sono impostati nell'elenco **ENUM_CALENDAR_EVENT_TYPE**

ID	Descrizione
CALENDAR_TYPE_EVENT	Evento (riunione, discorso, ecc.)
CALENDAR_TYPE_INDICATOR	Indicatore
CALENDAR_TYPE_HOLIDAY	Vacanza

Un settore dell'economia a cui un evento è correlato, è specificato nella struttura MqlCalendarEvent. I valori possibili sono impostati nell'elenco **ENUM_CALENDAR_EVENT_SECTOR**

ID	Descrizione
CALENDAR_SECTOR_NONE	Il settore non è impostato
CALENDAR_SECTOR_MARKET	Mercato, scambio
CALENDAR_SECTOR_GDP	Prodotto interno lordo (PIL)
CALENDAR_SECTOR_JOBS	Mercato del lavoro
CALENDAR_SECTOR_PRICES	Prezzi
CALENDAR_SECTOR_MONEY	Denaro
CALENDAR_SECTOR_TRADE	Trading
CALENDAR_SECTOR_GOVERNMENT	Governo
CALENDAR_SECTOR_BUSINESS	Business
CALENDAR_SECTOR_CONSUMER	Consumo
CALENDAR_SECTOR_HOUSING	Alloggiamento
CALENDAR_SECTOR_TAXES	Tasse
CALENDAR_SECTOR_HOLIDAYS	Vacanze

L'importanza dell'evento è specificata nella struttura MqlCalendarEvent. I valori possibili sono impostati nell'elenco **ENUM_CALENDAR_EVENT_IMPORTANCE**

ID	Descrizione
CALENDAR_IMPORTANCE_NONE	L'importanza non è impostata
CALENDAR_IMPORTANCE_LOW	Poca importanza
CALENDAR_IMPORTANCE_MODERATE	Media importanza
CALENDAR_IMPORTANCE_HIGH	Alta importanza

Il tipo di unità di misura utilizzato nella visualizzazione dei valori degli eventi è specificato nella struttura MqlCalendarEvent. I valori possibili sono impostati nell'elenco **ENUM_CALENDAR_EVENT_UNIT**

ID	Descrizione
CALENDAR_UNIT_NONE	L'unità di misura non è impostata
CALENDAR_UNIT_PERCENT	Percentuale
CALENDAR_UNIT_CURRENCY	Moneta nazionale
CALENDAR_UNIT_HOUR	Ore
CALENDAR_UNIT_JOB	Lavori

ID	Descrizione
CALENDAR_UNIT_RIG	Impianti di perforazione
CALENDAR_UNIT_USD	USD
CALENDAR_UNIT_PEOPLE	Persone
CALENDAR_UNIT_MORTGAGE	Mutui
CALENDAR_UNIT_VOTE	voti
CALENDAR_UNIT_BARREL	Barili
CALENDAR_UNIT_CUBICFEET	Piedi cubici
CALENDAR_UNIT_POSITION	Posizioni nette non commerciali
CALENDAR_UNIT_BUILDING	Edifici

In alcuni casi, i valori dei parametri economici richiedono un moltiplicatore impostato nella struttura MqlCalendarEvent. I possibili valori moltiplicatori sono impostati nella lista **ENUM_CALENDAR_EVENT_MULTIPLIER**

ID	Descrizione
CALENDAR_MULTIPLIER_NONE	Il moltiplicatore non è impostato
CALENDAR_MULTIPLIER_THOUSANDS	Migliaia
CALENDAR_MULTIPLIER_MILLIONS	Milioni
CALENDAR_MULTIPLIER_BILLIONS	Miliardi
CALENDAR_MULTIPLIER_TRILLIONS	Triliardi

Il potenziale impatto dell'evento su un tasso di cambio nazionale è indicato nella struttura MqlCalendarValue. I valori possibili sono impostati nell'elenco **ENUM_CALENDAR_EVENT_IMPACT**

ID	Descrizione
CALENDAR_IMPACT_NA	L'impatto non è impostato
CALENDAR_IMPACT_POSITIVE	Impatto positivo
CALENDAR_IMPACT_NEGATIVE	Impatto negativo

L'orario dell'evento è specificato nella struttura MqlCalendarEvent. I valori possibili sono impostati nell'elenco **ENUM_CALENDAR_EVENT_TIMEMODE**

ID	Descrizione
CALENDAR_TIMEMODE_DATETIME	La fonte pubblica un orario esatto di un evento

ID	Descrizione
CALENDAR_TIMEMODE_DATE	L'evento dura tutto il giorno
CALENDAR_TIMEMODE_NOTIME	La fonte non pubblica l'orario di un evento
CALENDAR_TIMEMODE_TENTATIVE	La fonte pubblica il giorno di un evento piuttosto che il suo orario esatto. L'orario è specificato al verificarsi dell'evento.

Guarda anche

Calendario Economico

I codici di Errore e di Avvertenza

Questa sezione contiene le seguenti descrizioni:

- Codici di ritorno del trade server - analisi dei risultati di richieste trade dalla funzione OrderSend();
- Avvisi del compilatore - codici di messaggi di avviso che vengono visualizzati in fase di compilazione (non errori);
- Errori di compilazione - i codici dei messaggi di errore ad un tentativo non riuscito di compilazione;
- Runtime errors – codici errore nell'esecuzione di programmi mql5 che possono essere ottenuti usando la funzione GetLastError().

Codici di Ritorno Trade Server

Tutte le richieste per eseguire le operazioni di trade vengono inviate come una struttura di una richiesta di trade MqlTradeRequest usando la funzione OrderSend(). Il risultato dell'esecuzione della funzione è posto alla struttura MqlTradeResult, il cui campo *retcode* contiene codice di ritorno del trade server.

Codice	Constant	Descrizione
10004	TRADE_RETCODE_REQUOTE	Riquotazione
10006	TRADE_RETCODE_REJECT	Richiesta rigettata
10007	TRADE_RETCODE_CANCEL	Richiesta annullata dal trader
10008	TRADE_RETCODE_PLACED	Ordine piazzato
10009	TRADE_RETCODE_DONE	Richiesta completata
10010	TRADE_RETCODE_DONE_PARTIAL	Solo una parte della richiesta è stata completata
10011	TRADE_RETCODE_ERROR	Errore elaborazione richiesta
10012	TRADE_RETCODE_TIMEOUT	Richiesta annullata per timeout
10013	TRADE_RETCODE_INVALID	Richiesta non valida
10014	TRADE_RETCODE_INVALID_VOLUME	Volume non valido nella richiesta
10015	TRADE_RETCODE_INVALID_PRICE	Prezzo non valido nella richiesta
10016	TRADE_RETCODE_INVALID_STOPS	Stops non validi nella richiesta
10017	TRADE_RETCODE_TRADE_DISABLED	Il trade è disattivato
10018	TRADE_RETCODE_MARKET_CLOSED	Il mercato è chiuso
10019	TRADE_RETCODE_NO_MONEY	Non ci sono abbastanza soldi per completare la richiesta
10020	TRADE_RETCODE_PRICE_CHANGED	Prezzi cambiato
10021	TRADE_RETCODE_PRICE_OFF	Non ci sono quotazioni per elaborare la richiesta
10022	TRADE_RETCODE_INVALID_EXPIRATION	Data di espirazione dell'ordine non valida nella richiesta
10023	TRADE_RETCODE_ORDER_CHANGED	Stato ordine cambiato
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	Richieste troppo frequenti
10025	TRADE_RETCODE_NO_CHANGES	Nessuna modifica nella richiesta
10026	TRADE_RETCODE_SERVER_DISABLES_AT	Autotrading disabilitato dal server
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	Autotrading disabilitato dal terminale client

Codice	Constant	Descrizione
10028	TRADE_RETCODE_LOCKED	Richiesta bloccata pe l'elaborazione
10029	TRADE_RETCODE_FROZEN	Ordine o posizione congelati
10030	TRADE_RETCODE_INVALID_FILL	Non valido tipo di filling dell'ordine
10031	TRADE_RETCODE_CONNECTION	Nessuna connessione con il trade server
10032	TRADE_RETCODE_ONLY_REAL	Il funzionamento è consentito solo per i conti live
10033	TRADE_RETCODE_LIMIT_ORDERS	Il numero di ordini in corso ha raggiunto il limite
10034	TRADE_RETCODE_LIMIT_VOLUME	Il volume degli ordini e posizioni per il simbolo ha raggiunto il limite
10035	TRADE_RETCODE_INVALID_ORDER	Errato o proibitotipo di ordine
10036	TRADE_RETCODE_POSITION_CLOSED	La posizione con lo specificato POSITION_IDENTIFIER è stata già chiusa
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	Il volume di chiusura supera il volume della posizione corrente
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	Un ordine già chiuso esiste già per una posizione specificata. Questo può accadere quando si lavora nel sistema hedging: • quando si tenta di chiudere una posizione con quella opposta, mentre esistono già ordini di chiusura per la posizione • quando si tenta di chiudere completamente o parzialmente una posizione se il volume totale dei già presenti ordini di chiusura ed i nuovi piazzati eccedono il volume della posizione corrente
10040	TRADE_RETCODE_LIMIT_POSITIONS	l numero di posizioni aperte contemporaneamente presenti sull'account può essere limitato dalle impostazioni del server. Una volta raggiunto il limite, il server restituisce l'errore TRADE_RETCODE_LIMIT_POSITIONS quando si tenta di piazzare un ordine. La limitazione funziona in modo diverso a seconda del tipo di accounting della posizione: • Netting(compensazione) — viene considerato il numero di posizioni

Codice	Constant	Descrizione
		aperte. Quando viene raggiunto un limite, la piattaforma disabilita il piazzamento di nuovi ordini la cui esecuzione può aumentare il numero di posizioni aperte. Infatti, la piattaforma permette di piazzare ordini solo per i simboli che già hanno posizioni aperte. Gli ordini pendenti attuali non sono considerati in quanto la loro esecuzione può portare a cambiamenti nelle posizioni attuali, ma non può aumentare il loro numero. • Hedging(copertura) — gli ordini pendenti vengono considerati insieme alle posizioni aperte, dal momento che l'attivazione id un ordine pendente porta sempre ad aprire una nuova posizione. Quando viene raggiunto un limite, la piattaforma disabilita sia i nuovi ordini di mercato per l'apertura di posizioni che gli ordini pendenti.
10041	TRADE_RETCODE_REJECT_CANCEL	La richiesta di attivazione dell'ordine pendente viene rifiutata, l'ordine viene annullato
10042	TRADE_RETCODE_LONG_ONLY	La richiesta è stata respinta perché è impostata per il simbolo la regola "**Sono consentite solo posizioni long**" (POSITION_TYPE_BUY)
10043	TRADE_RETCODE_SHORT_ONLY	La richiesta è stata respinta perché è impostata per il simbolo la regola "**Sono consentite solo posizioni short**" (POSITION_TYPE_SELL)
10044	TRADE_RETCODE_CLOSE_ONLY	La richiesta è stata respinta perché è impostata per il simbolo la regola "**È consentita solo la chiusura della posizione**"
10045	TRADE_RETCODE_FIFO_CLOSE	La richiesta è stata respinta perché è impostato per l'account di trading il flag "**La chiusura della posizione è consentita solo dalla regola FIFO**" (ACCOUNT_FIFO_CLOSE=true)

Avvisi del compilatore

Avvisi del compilatore sono indicati solo a scopo informativo e non sono i messaggi di errore.

Codice	Descrizione
21	Record incompleto di una data nella stringa datetime
22	Numero errato nella stringa datetime per la data. Requisiti: Anno 1970 <= X <= 3000 Mese 0 <X <= 12 Giorno 0 <X <= 31/30/28 (29)....
23	Numero errato di stringa datetime per l'orario. Requisiti: Ora 0 <= X <24 Minuto 0 <= X <60
24	Colori non validi in formato RGB: uno dei componenti RGB è minore di 0 o maggiore di 255
25	Carattere sconosciuto delle sequenze di escape. Conosciute: \n \r \t \\ \" \' \X \x
26	Volume troppo elevato di variabili locali (> 512Kb) della funzione, ridurre il numero
29	Enumerazione già definita (duplicazione) - i membri verranno aggiunti alla prima definizione
30	Overriding macro
31	La variabile viene dichiarata ma non è utilizzato da nessuna parte
32	Il costruttore deve essere di tipo void
33	Il distruttore deve essere di tipo void
34	La costante non rientra nel campo di numeri interi (X> _UI64_MAX | | X <_I64_MIN) e sarà convertita nel tipo double
35	HEX Troppo a lungo - più di 16 caratteri significativi (vengono tagliati gli spunti alti)
36	Non ci sono spunti in HEX stringa "0x"
37	Nessuna funzione - nulla deve essere eseguito
38	Una variabile non inizializzata viene usata
41	La funzione non ha corpo, e non è chiamata
43	Possibile perdita di dati al typecasting. Esempio: int x = (double) z;
44	Perdita di precisione (dei dati) per la conversione di una costante. Esempio: int x = M_PI
45	La differenza tra i segni degli operandi nelle operazioni di confronto. Esempio: (char) c1> (uchar) c2

Codice	Descrizione
46	Problemi con la funzione di importazione - la dichiarazione di #import è richiesta o l'importazione di funzioni è chiusa
47	Descrizione troppo grande - i caratteri aggiuntivi non saranno inclusi nel file eseguibile
48	Il numero di buffer indicatori dichiarati è inferiore a quello richiesto
49	Nessun colore per tracciare una serie grafica dell'indicatore
50	Nessuna serie grafica per disegnare l'indicatore
51	Handler 'OnStart' non trovato nello script
52	Handler 'OnStart' viene definito con parametri errati
53	La funzione 'OnStart' può essere definita solo in uno script
54	La funzione 'OnInit' viene definita con parametri errati
55	La funzione 'OnInit' non viene utilizzata negli script
56	La funzione 'OnDeinit' viene definita con parametri errati
57	La funzione 'OnDeinit' non viene utilizzata negli script
58	Due funzioni 'OnCalculate' vengono definite. OnCalculate() in un array prezzo verrà utilizzata
59	Eccessivo riempimento rilevato nel calcolo di un complesso numero intero costante
60	Probabilmente, la variabile è non inizializzata.
61	Questa dichiarazione rende impossibile vedere la variabile locale dichiarata sulla riga specificata
62	Questa dichiarazione rende impossibile vedere la variabile globale dichiarata sulla riga specificata
63	Non può essere usato per array statico allocato
64	Questo dichiarazione delle variabili nasconde una variabile predefinita
65	Il valore dell'espressione è sempre true/false
66	Utilizzando una variabile o espressione di tipo bool nelle operazioni matematiche è un rischio
67	Il risultato dell'applicazione dell'operatore unario meno per un tipo unsigned ulong è indefinito
68	La versione specificata in #property version è inaccettabile per il Marketsezione; il formato corretto di #property version id "XXX.YYY"
6	Trovata dichiarazione controllata vuota
70	Non valido tipo di ritorno o parametri non corretti durante la dichiarazione della funzione event handler

Codice	Descrizione
71	Un implicito cast di strutture per un tipo è richiesto
72	Questa dichiarazione rende impossibile l'accesso diretto al membro di una classe dichiarato nella stringa specificata. L' accesso sarà possibile solo con l' operazione di risoluzione ambito ::
73	La costante binaria è troppo grande, alte cifre ordine verranno troncate
74	Parametro nel metodo della classe ereditata ha una diverso modificatore const, la funzione derivata ha stracaricato(overloadato) la funzione padre
75	Valori di slittamento negativi o troppo grandi inoperazione di slittamento bit a bit, hanno un risultato dell'esecuzione non definito
76	Le funzioni devono restituire un valore
77	le funzioni void restituiscono un valore
78	Non tutti i percorsi di controllo restituiscono un valore
79	Le espressioni non sono consentite su scala globale
80	Controllare la precedenza degli operatori per un possibile errore; usare le parentesi per chiarificare la precedenza
81	Due OnCalCulate() vengono definite. Verrà usata la versione OHLC
82	Struct non ha memri, grandezza assegnata ad 1 byte
83	I valori di ritorno della funzione devono essere controllati
84	Resource indicator is compiled for debugging. That slows down the performance. Please recompile the indicator to increase performance
85	Too great character code in the string, must be in the range 0 to 65535
86	Unrecognized character in the string
87	No indicator window property (setting the display in the main window or a subwindow) is defined. Property #property indicator_chart_window is applied

Errori di compilazione

MetaEdtior 5 mostra i messaggi di errore sugli errori rilevati dal programma dal compilatore built-in durante la compilazione. L'elenco di questi errori è riportato qui sotto nella tabella. Per compilare il codice sorgente in un unico eseguibile, premere **F7**. I programmi che contengono errori non possono essere compilati fino a quando gli errori individuati dal compilatore non vengono eliminati.

Codice	Descrizione
100	Errore lettura file
101	Errore di *. Apertura file EX5 in scrittura mentre si salva
103	Memoria insufficiente per completare la compilazione
104	Unità sintattica vuota non riconosciuta dal compilatore
105	Nome di file non corretto in #include
106	Errore durante l'accesso ad un file #include (probabilmente il file non esiste)
108	Nome inappropriato per #define
109	Comando sconosciuto del preprocessore (valido #include, #define, #proprietà #import)
110	Simbolo sconosciuto al compilatore
111	Funzione non implementata (la descrizione è presente, ma nessun corpo)
112	Doppie virgolette (") omesse
113	Parentesi angolare di apertura (<) o doppie virgolette (") omesse
114	Virgoletta singola (') omessa
115	Parentesi angolare di chiusura ">" omessa
116	Tipo non specificato nella dichiarazione
117	Nessun operatore o ritorno è stato trovato in nessuno di tutti i rami dell'implementazione
118	Si aspettava la parentesi di apertura della chiamata dei parametri
119	Errore durante la scrittura EX5
120	Accesso non valido ad un array
121	La funzione non è di tipo void e l'operatore return deve restituire un valore
122	Dichiarazione errata del distruttore
123	Due punti ":" Manca
124	La variabile è già dichiarata
125	La variabile con tale identificatore già dichiarata
126	Nome della variabile troppo lungo (> 250 caratteri)

Codice	Descrizione
127	Struttura con tale identificatore già definita
128	La struttura non è definita
129	Membro della struttura con lo stesso nome già definito
130	Nessun membro di tale struttura
131	Accoppiamento violato di parentesi
132	Attesa apertura di parentesi "("
133	Parentesi sbilanciate (nessun "}")
134	Difficile nel compilare (troppa ramificazione, livelli di stack interni sono troppo pieni)
135	Errore di apertura del file per la lettura
136	Memoria insufficiente per scaricare il file di origine in memoria
13	E' attesa una variabile
138	La referenza non può essere inizializzata
140	Assegnazione attesa (appare nella dichiarazione)
141	Attesa apertura di parentesi "{"
142	Il parametro può essere un array dinamico solo
143	L'uso del tipo "void" è inaccettabile
144	Nessuna coppia per ")" o "]", cioè "(" o "[" è assente
145	Nessuna coppia per "(" o "{", cioè ")" o "]" è assente
146	Grandezza array errata
147	Troppi parametri (> 64)
149	Questo token non è previsto qui
150	Utilizzo non valido dell'operazione (operandi non validi)
151	Espressione di tipo void non consentita
152	E' atteso un operatore
153	Misuso di break
154	Atteso un punto e virgola ";"
155	Attesa una virgola ","
156	Deve essere un tipo della classe, non struct
157	Attesa un espressione

Codice	Descrizione
158	"carattere non HEX" trovato in HEX o un numero troppo lungo (numero di cifre > 511)
159	La costante-stringa ha più di 65534 caratteri
160	La definizione di funzione è inaccettabile qui
161	Fine del programma inattesa
162	Dichiarazione in avanti, è vietata per le strutture
163	La funzione con questo nome è già definita ed ha un altro tipo di ritorno
164	La funzione con questo nome è già definita ed ha un diverso insieme di parametri
165	La funzione con questo nome è già definita ed implementata
166	Funzione sovraccaricata per questa chiamata, non è stata trovata
167	Funzione con un valore di ritorno di tipo void non può restituire un valore
168	La funzione non è definita
170	Valore atteso
171	Nell' espressione *case* solo costanti integer sono valide
172	Il valore di *case* in questo *switch* è già usato
173	Atteso un Integer
174	Nell' espressione #import è atteso un nome del file
175	Le espressioni non sono ammesse a livello globale
176	Omessa parentesi ")" prima di ";"
177	A sinistra del segno di uguaglianza, è attesa una variabile
178	Il risultato di espressione non è utilizzato
179	La dichiarazione di variabili non è consentita in *case*
180	Conversione implicita da una stringa in un numero
181	Conversione implicita di un numero in una stringa
182	Chiamata ambigua di una funzione sovraccaricata (diversi overloads riempiti)
183	Illegale *else* senza un adeguato *if*
184	Non valido *case* o *default* senza *switch*
185	Uso inappropriato di puntini di sospensione
186	La sequenza di inizializzazione ha più elementi rispetto alla variabile inizializzata
187	Una costante per *case* attesa
188	Richiesta una costante espressione

Codice	Descrizione
189	Una variabile costante non può essere modificata
190	Chiusura di parentesi o una virgola, è prevista (dichiarazione di membri array)
191	Identificatore Enumeratore già definito
192	L' enumerazione non può avere modificatori di accesso (const, extern, static)
193	Membro di enumerazione già dichiarato con un valore diverso
194	C'è una variabile definita con lo stesso nome
195	C'è una struttura definita con lo stesso nome
196	Nome del membro di enumerazione, atteso
197	Espressione Integer attesa
198	Divisione per zeo, in un'espressione costante
199	Numero errato di parametri nella funzione
200	Il parametro di riferimento deve essere una variabile
201	Variabile dello stesso tipo da passare per riferimento atteso
202	Una variabile costante non può essere passata da un riferimento non-costante
203	Richiede un intero positivo costante
204	Impossibile accedere al membro classe protected
205	Import già definito in un altro modo
208	File eseguibile non creato
209	'OnCalculate entry point, non trovato per l'indicatore
210	L'operazione continue può essere utilizzata solo all'interno di un ciclo
211	Errore di accesso ad un membro della classe private (chiuso)
213	Il Metodo di struttura o classe non è dichiarato
214	Errore di accesso ad un metodo della classe private (chiuso)
216	La copia di strutture con oggetti non è consentita
218	Indice fuori intervallo dell'array
219	Inizializzazione dell'array nella struttura o nella dichiarazione della classe non è ammessa
220	Costruttore della classe non può avere parametri
221	Distruttore della classe non può avere parametri
222	Metodo della classe o una struttura con lo stesso nome e parametri sono già stati dichiarati

Codice	Descrizione
223	Operando atteso
224	Metodo della classe o struttura con lo stesso nome, ma con parametri diversi (dichiarazione !=implementazione)
225	La funzione importata non è descritta
226	ZeroMemory() is not allowed for objects with protected members or inheritance
227	Chiamata ambigua della funzione sovraccaricata (corrispondenza esatta dei parametri per diversi overloads)
228	Nome variabile atteso
229	Un riferimento non può essere dichiarato in questo punto
230	Già utilizzato come nome dell'enumerazione
232	Classe o struttura attesa
235	Impossibile chiamare l' operatore 'delete' per cancellare l'array
236	Operatore 'while' atteso
237	L' operatore 'delete' deve avere un puntatore
238	Vi è 'default' per questo 'switch' già
239	Errore di sintassi
240	Sequenza-escape può avvenire solo nelle stringhe (inizia con '\')
241	Array richiesto - la parentesi quadra '[' non si applica ad un array, o non array vengono passati come parametri di array
242	Non può essere inizializzato attraverso la sequenza di inizializzazione
243	Import non è definito
244	Errore Ottimizzatore sull'albero sintattico
245	Dichiarate troppe strutture (cerca di semplificare il programma)
246	La conversione del parametro non è consentita
247	Uso improprio dell' operatore 'delete'
248	Non è permesso di dichiarare un puntatore ad un riferimento
249	Non è permesso di dichiarare un riferimento ad un riferimento
250	Non è permesso di dichiarare un puntatore ad un puntatore
251	La dichiarazione di struttura nella lista dei parametri non è consentita
252	Operazione di typecasting non valida
253	Un puntatore può essere dichiarato solo per una classe o una struttura
256	Identificatore non dichiarato

Codice	Descrizione
257	Errore ottimizzatore codice eseguibile
258	Errore generazione codice eseguibile
260	Espressione non valida per l'operatore 'switch'
261	Pool di costanti stringa sovrariempite, semplificare programma
262	Non è possibile convertire l'enumerazione
263	Non usare 'virtual' per i dati (i membri di una classe o di una struttura)
264	Impossibile chiamare il metodo protected della classe
265	Funzioni virtuali sovraccaricate restituiscono un tipo differente
266	La classe non può essere ereditata da una struttura
267	La struttura non può essere ereditata da una classe
268	Costruttore non può essere virtuale (specificatore *virtual* non consentito)
269	La struttura non può avere metodi virtuali
270	La funzione deve avere un corpo
271	L'overloading di funzioni di sistema (funzioni del terminale) è vietato
272	*Const* (specificatore) non è valido per le funzioni che non sono membri di una classe o di una struttura
274	Non è permesso cambiare i membri della classe nel metodo costante
276	Sequenza di inizializzazione inappropriata
277	Mancato valore default per il parametro (dichiarazione specifica di parametri di default)
278	Overriding del parametro di default (valori diversi nella dichiarazione ed implementazione)
279	Non è permesso chiamare un metodo non-costante per un oggetto costante
280	Un oggetto è necessario per l'accesso ai membri (è impostato un punto per una non classe/struttura)
281	Il nome di una struttura già dichiarata non può essere utilizzato in una dichiarazione
284	Conversione non autorizzata (in eredità chiusa)
285	Strutture ed array non possono essere utilizzate come variabili di input
286	*Const* (specificatore) non è valido per il costruttore/distruttore
287	Stringa di espressione non corretta per un valore datetime
288	Proprietà sconosciuta (#property)

Codice	Descrizione
289	Valore errato di una proprietà
290	Indice non valido di una proprietà in #property
291	Omessa chiamata parametro - <func (x,)>
293	Gli oggetti devono essere passati per riferimento
294	Gli array devono essere passati per riferimento
295	La funzione è stata dichiarata come esportabile
296	La funzione non è stata dichiarata come esportabile
297	E 'vietata l'esportazione di una funzione importata
298	La funzione importata non può avere questo parametro (divieto di passare un puntatore, classe o una struttura che contiene un array dinamico, puntatore, classe, ecc)
299	Deve essere una classe
300	#import non è stato chiuso
302	Tipo non corrispondente
303	Variabile Extern già inizializzata
304	Non sono stati trovati nessuna funzione esportata o entry point
305	Chiamata al costruttore esplicito non è consentita
306	Il metodo è stato dichiarato come costante
307	Il metodo non è stato dichiarato come costante
308	Grandezza non corretta del file di risorse
309	Nome di risorsa non corretto
310	Errore di apertura di file risorsa
311	Errore di lettura di file risorsa
312	Risorsa di tipo sconosciuto
313	Percorso corretto al file di risorse
314	Il nome della risorsa specificata è già utilizzato
315	Argomento previsto per la funzione-tipo macro
316	Simbolo imprevisto in definizione macro
317	Errore nei parametri formali della macro
318	Numero non valido di parametri per una macro
319	Troppi parametri per una macro

Codice	Descrizione
320	Troppo complesso, semplificare la macro
321	Parametro per EnumToString() può essere solo una enumerazione
322	Il nome risorsa è troppo lungo
323	Formato immagine non supportato (solo BMP con 24 o 32 bit di profondità colore sono supportati)
324	Un array non può essere dichiarato in un operatore
325	La funzione può essere dichiarata solo nell'ambito globale
326	La dichiarazione non è consentita per il corrente ambito
327	L'inizializzazione di variabili statiche con i valori delle variabili locali non è consentito
328	Dichiarazione illegale di un array di oggetti che non hanno un costruttore di default
329	Lista inizializzazione consentita solo per costruttori
330	Nessuna definizione di funzione dopo l'elenco di inizializzazione
331	La lista di Inizializzazione è vuota
332	Inizializzazione di array in un costruttore non è consentita
333	Inizializzazione membri di una classe genitore nella lista inizializzazione non è consentita
334	Espressione di tipo intero attesa
335	La memoria richiesta per l' array supera il valore massimo
336	La memoria richiesta per la struttura supera il valore massimo
337	La memoria richiesta per le variabili dichiarate sul livello globale supera il valore massimo
338	La memoria richiesta per variabili locali supera il valore massimo
339	Costruttore non definito
340	Non valido il nome del file icona
341	Impossibile aprire il file iconanel percorso specificato
342	Il file icona non è corretto e non è del formato ICO(formato)
343	La reinizializzazione di un membro in un costruttore di una classe/struttura usando la lista inizializzazione
344	Inizializzazione di membri static nella lista inizializzazione del costruttore, non è consentita
345	Inizializzazione di un membro non-statico di una classe/struttura su un livello globale non è consentito

Codice	Descrizione
346	Il nome del metodo della classe/struttura corrisponde al nome di un membro dichiarato in precedenza
347	Il nome del membro della classe/struttura corrisponde al nome di un metodo dichiarato in precedenza
348	Virtual (funzione) non può essere dichiarata come static
349	Il modificatore const non è consentito per le funzioni static
350	Il costruttore o distruttore non possono essere static
351	Membro/metodo non-static di una classe o di una struttura non possono essere acceduti da una funzione static
352	Un'operazione di overload (+,-,[],++,-- ecc.) è attesa dopo l' operatore keyword
353	Non tutte le operazioni possono essere sovraccaricate in MQL5
354	La definizione non corrisponde alla dichiarazione
355	Un numero non valido di parametri è specificato per l' operatore
356	Funzione di event handlingv non trovata
357	Il metodo non può essere esportato
358	Un puntatore all'oggetto costante non può essere normalizzato da un oggetto non-costante
359	I templated della classe non sono ancora supportati
360	L' overload dei template della funzione, non è ancora supportato
361	Template della funzione non può essere applicato
362	Parametro ambiguo nel template della funzione (diversi tipi di parametri possono essere applicati)
363	Impossibile determinare il tipo di parametro, con la quale l'argomento del modello della funzione deve essere normalizzato
364	Numero errato di parametri del modello della funzione
365	Il modello della funzione non può essere virtual
366	Templates della funzione non possono essere esportati
367	Templated di funzioni non possono essere importati
368	Strutture che contengono gli oggetti non sono ammesse
369	Array e strutture contenenti gli oggetti String non sono ammessi
370	Un membro stato di classe/struttura dev'essere esplicitamente inizializzato
371	Limitazione compilatore: la stringa non può contenere più di 65 535 caratteri
37	Inconsistent #ifdef/#endif

Codice	Descrizione
373	L' oggetto della classe non può essere restituito, costruttore di copia non trovato
374	Membri e metodi non statici non possono essere utilizzati
375	OnTesterInit() impossibile da usare senza OnTesterDeinit()
376	Redefinition of formal parameter '%s'
377	Macro __FUNCSIG__ and __FUNCTION__ non può apparire al di fuori del corpo di una funzione
378	Tipo restituito non valido. Ad esempio, questo errore verrà prodotto per le funzioni importate dalla DLL che restituiscono struttura o puntatore.
379	Errore d'uso template
380	Non usato
381	Sintassi illegale quando si dichiara una funzione virtuale pura, solo "=NULL" o "=0" sono consentiti
382	Solo le funzioni virtuali possono essere dichiarate con lo specificatore-puro ("=NULL" or "=0")
383	La classe astratta non può essere instanziata
384	Un puntatore a un tipo definito dall'utente deve essere applicato come tipo di destinazione per il cast dinamico utilizzando l'operatore dynamic_cast
385	È previsto il tipo "Puntatore a funzione"
386	I puntatori ai metodi non sono supportati
387	Errore - impossibile definire il tipo di puntatore per la funzione
388	Il tipo di cast non è disponibile a causa dell' eredità private
389	Una variabile con il modificatore const deve essere inizializzata durante la dichiarazione
393	Solo metodi con l'accesso public possono essere dichiarati in un' interfaccia
394	Annidamento non valido di un' interfaccia all'interno di un'altra interfaccia
395	Un'interfaccia può essere derivata solo da un'altra interfaccia
396	Un interfaccia è prevista
397	Le interfacce supportano solo l'eredità public
398	Un interfaccia non può contenere membri
399	Gli oggetti dell'interfaccia non possono essere creati direttamente, utilizzare solo l'ereditarietà
400	Uno specificatore non può essere utilizzato in una dichiarazione forward

Codice	Descrizione
401	L'ereditarietà della classe è impossibile, poiché è dichiarata con lo specificatore final
402	Non posso ridefinire un metodo dichiarato con lo specificatore final
403	Lo specificatore final può essere applicato solo alle funzioni virtual
404	Il metodo contrassegnato dallo specificatore override in realtà non sovrascrive alcuna funzione della classe base
405	Uno specificatore non è autorizzato a definire una funzione, ma solo nel dichiarare
406	Impossibile castare il tipo verso quello specificato
407	Il tipo non può essere utilizzato per una risorsa variabile
408	Errore nel file del progetto
409	Non può essere usato come un membro union(unione)
410	Scelta ambigua per il nome, il contesto di utilizzo deve essere definito in modo esplicito
411	La struttura non può essere utilizzata dalla DLL
412	Impossibile chiamare una funzione contrassegnata dallo specificatore delete
413	MQL4 non supportato. Per compilare il programma, utilizzare MetaEditor nella cartella di installazione del terminale MetaTrader 4

Errori di Runtime

GetLastError() è la funzione che restituisce l'ultimo codice di errore memorizzato nella variabile predefinita _LastError. Questo valore può essere azzerato dalla funzione ResetLastError().

Costante	Codice	Descrizione
ERR_SUCCESS	0	L'operazione è stata completata con successo
ERR_INTERNAL_ERROR	4001	Errore interno imprevisto
ERR_WRONG_INTERNAL_PARAMETER	4002	Parametri non validi nella chiamata interna della funzione del terminale client
ERR_INVALID_PARAMETER	4003	Parametro errato quando si chiama la funzione di sistema
ERR_NOT_ENOUGH_MEMORY	4004	Memoria insufficiente per eseguire la funzione del sistema
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	La struttura contiene oggetti di stringhe e/o array dinamici e/o la struttura di tali oggetti e/o classi
ERR_INVALID_ARRAY	4006	Array di un tipo errato, grandezza errata, o un oggetto danneggiato di un array dinamico
ERR_ARRAY_RESIZE_ERROR	4007	Memoria insufficiente per il trasferimento di un array, o un tentativo di modificare la grandezza di un array statico
ERR_STRING_RESIZE_ERROR	4008	Memoria insufficiente per il trasferimento di stringa
ERR_NOTINITIALIZED_STRING	4009	Stringa non inizializzata
ERR_INVALID_DATETIME	4010	Data e/o ora non validi
ERR_ARRAY_BAD_SIZE	4011	La quantità totale di elementi nell'array non può superare 2147483647
ERR_INVALID_POINTER	4012	Puntatore errato
ERR_INVALID_POINTER_TYPE	4013	Tipo errato di puntatore
ERR_FUNCTION_NOT_ALLOWED	4014	La funzione di sistema non è autorizzata a chiamare
ERR_RESOURCE_NAME_DUPLICATED	4015	I nomi delle risorse dinamiche e statiche, corrispondono

Costante	Codice	Descrizione
ERR_RESOURCE_NOT_FOUND	4016	Risorse con questo nome non sono state trovate in EX5
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	Tipo di risorsa non supportata o la sua grandezza supera i 16 Mb
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	Il nome della risorsa supera i 63 caratteri
ERR_MATH_OVERFLOW	4019	E' avvenuto l'overflow quando si calcolava la funzione matematica
ERR_SLEEP_ERROR	4020	Fuori data di fine test dopo aver chiamato Sleep()
ERR_PROGRAM_STOPPED	4022	Test forzatamente interrotto dall'esterno. Ad esempio, ottimizzazione interrotta, finestra di test visuali chiusa o agente di testing interrotto
ERR_INVALID_TYPE	4023	Tipo non valido
ERR_INVALID_HANDLE	4024	Handle non valido
ERR_TOO_MANY_OBJECTS	4025	Pool di oggetti pieno
Chart		
ERR_CHART_WRONG_ID	4101	ID del Chart errato
ERR_CHART_NO_REPLY	4102	Il chart non risponde
ERR_CHART_NOT_FOUND	4103	Chart non trovato
ERR_CHART_NO_EXPERT	4104	Nessun Expert Advisor nel chart che potrebbe gestire l'evento
ERR_CHART_CANNOT_OPEN	4105	Errore di apertura del chart
ERR_CHART_CANNOT_CHANGE	4106	Impossibile modificare il simbolo ed il periodo del chart
ERR_CHART_WRONG_PARAMETER	4107	Valore dell'errore dei parametri per la funzione di lavoro con i charts
ERR_CHART_CANNOT_CREATE_TIMER	4108	Impossibile creare il timer
ERR_CHART_WRONG_PROPERTY	4109	ID proprietà del chart, sbagliato
ERR_CHART_SCREENSHOT_FAILED	4110	Errore durante la creazione di uno screenshot
ERR_CHART_NAVIGATE_FAILED	4111	Errore di navigazione attraverso il grafico

Costante	Codice	Descrizione
ERR_CHART_TEMPLATE_FAILED	4112	Errore durante l'applicazione del template
ERR_CHART_WINDOW_NOT_FOUND	4113	La sottofinestra contenente l'indicatore non è stata trovata
ERR_CHART_INDICATOR_CANNOT_ADD	4114	Errore durante l'aggiunta di un indicatore al chart
ERR_CHART_INDICATOR_CANNOT_DEL	4115	Errore durante l'eliminazione di un indicatore dal chart
ERR_CHART_INDICATOR_NOT_FOUND	4116	Indicatore non trovato sul chart specificato
Oggetti grafici		
ERR_OBJECT_ERROR	4201	Errore di lavoro con un oggetto grafico
ERR_OBJECT_NOT_FOUND	4202	L' oggetto grafico non è stato trovato
ERR_OBJECT_WRONG_PROPERTY	4203	ID errato di una proprietà oggetto grafico
ERR_OBJECT_GETDATE_FAILED	4204	Impossibile ottenere data corrispondente al valore
ERR_OBJECT_GETVALUE_FAILED	4205	Impossibile ottenere il valore corrispondente alla data
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	Simbolo sconosciuto
ERR_MARKET_NOT_SELECTED	4302	Il simbolo non è selezionato in MarketWatch
ERR_MARKET_WRONG_PROPERTY	4303	Identificatore errato di una proprietà del simbolo
ERR_MARKET_LASTTIME_UNKNOWN	4304	L'orario dell'ultimo tick non è noto (nessun ticks)
ERR_MARKET_SELECT_ERROR	4305	Errore durante l'aggiunta o l'eliminazione di un simbolo in MarketWatch
Accesso allo storico		
ERR_HISTORY_NOT_FOUND	4401	Storico richiesto non trovato
ERR_HISTORY_WRONG_PROPERTY	4402	ID errato della proprietà dello storico

Costante	Codice	Descrizione
ERR_HISTORY_TIMEOUT	4403	Timeout eccessivo nella richiesta dello storico
ERR_HISTORY_BARS_LIMIT	4404	Numero di barre richieste limitato dalle impostazioni terminale
ERR_HISTORY_LOAD_ERRORS	4405	Errori multipli durante il caricamento dello storico
ERR_HISTORY_SMALL_BUFFER	4407	L'array ricevente è troppo piccolo per conservare tutti i dati richiesti
Variabili_Globali		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	La variabile globale del terminale cliente non è stata trovata
ERR_GLOBALVARIABLE_EXISTS	4502	La variabile globale del terminale client con lo stesso nome esiste già
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	Le variabili globali non sono state modificate
ERR_GLOBALVARIABLE_CANNOTREAD	4504	Non posso leggere il file con i valori delle variabiili globali
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	Non posso scrivere il file con i valori delle variabiili globali
ERR_MAIL_SEND_FAILED	4510	Invio email fallito
ERR_PLAY_SOUND_FAILED	4511	Riproduzione del suono fallita
ERR_MQL5_WRONG_PROPERTY	4512	Identificatore errato della proprietà del programma
ERR_TERMINAL_WRONG_PROPERTY	4513	Identificatore errato della proprietà del terminale
ERR_FTP_SEND_FAILED	4514	Invio file tramite ftp fallito
ERR_NOTIFICATION_SEND_FAILED	4515	Fallimento nell'invio di una notifica
ERR_NOTIFICATION_WRONG_PARAMETER	4516	Parametro non valido per l'invio di una notifica - una stringa vuota o NULL è stato passato alla funzione SendNotification()
ERR_NOTIFICATION_WRONG_SETTINGS	4517	Impostazioni errate di notifiche nel terminale (ID non è specificato o l'autorizzazione non è impostata)

Costante	Codice	Descrizione
ERR_NOTIFICATION_TOO_FREQUENT	4518	Invio troppo frequente delle notifiche
I Buffers Indicatore Personalizzato		
ERR_BUFFERS_NO_MEMORY	4601	Memoria insufficiente per la distribuzione dei buffer indicatore
ERR_BUFFERS_WRONG_INDEX	4602	Errato indice buffer indicatore
Proprietà Indicatore Personalizzato		
ERR_CUSTOM_WRONG_PROPERTY	4603	ID errato della proprietà indicatore personalizzato
Account		
ERR_ACCOUNT_WRONG_PROPERTY	4701	Proprietà ID dell'account, sbagliata
ERR_TRADE_WRONG_PROPERTY	4751	Proprietà ID del trade, sbagliata
ERR_TRADE_DISABLED	4752	Trading da Expert Advisors vietato
ERR_TRADE_POSITION_NOT_FOUND	4753	Posizione non trovata
ERR_TRADE_ORDER_NOT_FOUND	4754	Ordine non trovato
ERR_TRADE_DEAL_NOT_FOUND	4755	Affare non trovato
ERR_TRADE_SEND_FAILED	4756	Richiesta di trade non riuscita
ERR_TRADE_CALC_FAILED	4758	Errore nel calcolare profitto o margine
Indicatori		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	Simbolo sconosciuto
ERR_INDICATOR_CANNOT_CREATE	4802	L' indicatore non può essere creato
ERR_INDICATOR_NO_MEMORY	4803	Memoria insufficiente per aggiungere l'indicatore
ERR_INDICATOR_CANNOT_APPLY	4804	L'indicatore non può essere applicato ad un altro indicatore
ERR_INDICATOR_CANNOT_ADD	4805	Errore durante l'applicazione di un indicatore al grafico
ERR_INDICATOR_DATA_NOT_FOUND	4806	Dati richiesti non trovati
ERR_INDICATOR_WRONG_HANDLE	4807	Handle indicatore, errato
ERR_INDICATOR_WRONG_PARAMETERS	4808	Numero errato di parametri durante la creazione di un

Costante	Codice	Descrizione
		indicatore
ERR_INDICATOR_PARAMETERS_MISSING	4809	Nessun parametro quando si crea un indicatore
ERR_INDICATOR_CUSTOM_NAME	4810	Il primo parametro dell'array deve essere il nome dell' indicatore personalizzato
ERR_INDICATOR_PARAMETER_TYPE	4811	Parametro di tipo non valido nell'array durante la creazione di un indicatore
ERR_INDICATOR_WRONG_INDEX	4812	Indice errato del buffer indicatore richiesto
Profondità di Mercato		
ERR_BOOKS_CANNOT_ADD	4901	Profondità di Mercato non può essere aggiunta
ERR_BOOKS_CANNOT_DELETE	4902	Profondità di Mercato non può essere rimossa
ERR_BOOKS_CANNOT_GET	4903	I dati dalla Profondità di Mercato non possono essere ottenuti
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	Errore nella sottoscrizione di ricezione di nuovi dati da Profondità di Mercato
Operazioni con i file		
ERR_TOO_MANY_FILES	5001	Più di 64 file non possono essere aperti contemporaneamente
ERR_WRONG_FILENAME	5002	Nome di file non validi
ERR_TOO_LONG_FILENAME	5003	Nome del file troppo lungo
ERR_CANNOT_OPEN_FILE	5004	Errore durante l'apertura del file
ERR_FILE_CACHEBUFFER_ERROR	5005	Memoria insufficiente per la cache di lettura
ERR_CANNOT_DELETE_FILE	5006	Errore durante l'eliminazione del file
ERR_INVALID_FILEHANDLE	5007	Un file con questo handle è stato chiuso, o non è stato proprio aperto
ERR_WRONG_FILEHANDLE	5008	File handle errato
ERR_FILE_NOTTOWRITE	5009	Il file deve essere aperto per la scrittura

Costante	Codice	Descrizione
ERR_FILE_NOTTOREAD	5010	Il file deve essere aperto per la lettura
ERR_FILE_NOTBIN	5011	Il file deve essere aperto come un file binario
ERR_FILE_NOTTXT	5012	Il file deve essere aperto come un testo
ERR_FILE_NOTTXTORCSV	5013	Il file deve essere aperto come un testo o CSV
ERR_FILE_NOTCSV	5014	Il file deve essere aperto come CSV
ERR_FILE_READERROR	5015	Errore lettura file
ERR_FILE_BINSTRINGSIZE	5016	La grandezza della stringa deve essere specificata, in quanto il file viene aperto come binario
ERR_INCOMPATIBLE_FILE	5017	Un file di testo deve essere per gli array di stringhe, per altri array - binario
ERR_FILE_IS_DIRECTORY	5018	Questo non è un file, questa è una directory
ERR_FILE_NOT_EXIST	5019	Il file non esiste
ERR_FILE_CANNOT_REWRITE	5020	Il file non può essere riscritto
ERR_WRONG_DIRECTORYNAME	5021	Nome directory errata
ERR_DIRECTORY_NOT_EXIST	5022	Directory non esiste
ERR_FILE_ISNOT_DIRECTORY	5023	Questo è un file, non una directory
ERR_CANNOT_DELETE_DIRECTORY	5024	La directory non può essere rimossa
ERR_CANNOT_CLEAN_DIRECTORY	5025	Impossibile cancellare la directory (probabilmente uno o più file sono bloccati e l'operazione di rimozione è fallita)
ERR_FILE_WRITEERROR	5026	Impossibile scrivere una risorsa in un file
ERR_FILE_ENDOFFILE	5027	Impossibile leggere il successivo pezzo di dati dal file CSV (FileReadString, FileReadNumber, FileReadDatetime, FileReadBool),

Costante	Codice	Descrizione
		finchè non viene raggiunta la fine del file
Casting delle stringhe		
ERR_NO_STRING_DATE	5030	Nessuna data nella stringa
ERR_WRONG_STRING_DATE	5031	Data errata nella stringa
ERR_WRONG_STRING_TIME	5032	Orario sbagliato nella stringa
ERR_STRING_TIME_ERROR	5033	Errore di conversione da stringa a data
ERR_STRING_OUT_OF_MEMORY	5034	Memoria insufficiente per la stringa
ERR_STRING_SMALL_LEN	5035	La lunghezza della stringa è inferiore al previsto
ERR_STRING_TOO_BIGNUMBER	5036	Numero troppo largo, più di ULONG_MAX
ERR_WRONG_FORMATSTRING	5037	Formato stringa non valido
ERR_TOO_MANY_FORMATTERS	5038	Quantità di specificatori di formato superiore ai parametri
ERR_TOO_MANY_PARAMETERS	5039	Quantità dei parametri più degli specificatori di formato
ERR_WRONG_STRING_PARAMETER	5040	Parametro danneggiato di tipo stringa
ERR_STRINGPOS_OUTOFRANGE	5041	Posizione al di fuori della stringa
ERR_STRING_ZEROADDED	5042	0 aggiunto alla fine della stringa, una operazione inutile
ERR_STRING_UNKNOWNTYPE	5043	Dati di tipo sconosciuto durante la conversione in una stringa
ERR_WRONG_STRING_OBJECT	5044	Oggetto della stringa, dannegiato
Operazioni con Arrays		
ERR_INCOMPATIBLE_ARRAYS	5050	Copia di array incompatibili. Array di stringhe possono essere copiati solo per array di stringhe, ed un array numerico - solo in un array numerico
ERR_SMALL_ASSERIES_ARRAY	5051	L'array ricevente viene dichiarato come AS_SERIES, ed è di grandezza insufficiente

Costante	Codice	Descrizione
ERR_SMALL_ARRAY	5052	Array troppo piccolo, la posizione di partenza è fuori dell'array
ERR_ZEROSIZE_ARRAY	5053	Un array di lunghezza zero
ERR_NUMBER_ARRAYS_ONLY	5054	Deve essere un array numerico
ERR_ONEDIM_ARRAYS_ONLY	5055	Deve essere un array mono-dimensionale
ERR_SERIES_ARRAY	5056	Timeseries non possono essere utilizzate
ERR_DOUBLE_ARRAY_ONLY	5057	Deve essere un array di tipo double
ERR_FLOAT_ARRAY_ONLY	5058	Deve essere un array di tipo float
ERR_LONG_ARRAY_ONLY	5059	Deve essere un array di tipo long
ERR_INT_ARRAY_ONLY	5060	Deve essere un array di tipo int
ERR_SHORT_ARRAY_ONLY	5061	Deve essere un array di tipo corto
ERR_CHAR_ARRAY_ONLY	5062	Deve essere un array di tipo char
ERR_STRING_ARRAY_ONLY	5063	Deve essere un array di tipo stringa
Operazioni con OpenCL		
ERR_OPENCL_NOT_SUPPORTED	5100	Funzioni OpenCL non sono supportate su questo computer
ERR_OPENCL_INTERNAL	5101	Un errore interno si è verificato quando si eseguiva OpenCL
ERR_OPENCL_INVALID_HANDLE	5102	Non valido handle OpenCL
ERR_OPENCL_CONTEXT_CREATE	5103	Errore durante la creazione del contesto OpenCL
ERR_OPENCL_QUEUE_CREATE	5104	Impossibile creare una coda di esecuzione in OpenCL
ERR_OPENCL_PROGRAM_CREATE	5105	Si è verificato un errore durante la compilazione di un programma OpenCL
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	Nome kernel troppo lungo (OpenCL kernel)
ERR_OPENCL_KERNEL_CREATE	5107	Errore durante la creazione di un OpenCL kernel
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	Errore avvenuto durante impostazione dei parametri per il

Costante	Codice	Descrizione
		kernel OpenCL
ERR_OPENCL_EXECUTE	5109	Errore programma OpenCL:errore di runtime
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	Dimensione non valida del buffer OpenCL
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	Offset non valido nel buffer OpenCL
ERR_OPENCL_BUFFER_CREATE	5112	Impossibile creare un buffer OpenCL
ERR_OPENCL_TOO_MANY_OBJECTS	5113	Troppi oggetti OpenCL
ERR_OPENCL_SELECTDEVICE	5114	Errore selezione dispositivo OpenCL
Working with databases		
ERR_DATABASE_INTERNAL	5120	Errore interno del database
ERR_DATABASE_INVALID_HANDLE	5121	Handle del database non valido
ERR_DATABASE_TOO_MANY_OBJECTS	5122	Superato il numero massimo accettabile di oggetti Database
ERR_DATABASE_CONNECT	5123	Errore di connessione al database
ERR_DATABASE_EXECUTE	5124	Errore esecuzione della Richiesta
ERR_DATABASE_PREPARE	5125	Errore generazione della Richiesta
ERR_DATABASE_NO_MORE_DATA	5126	Non ci sono più dati da leggere
ERR_DATABASE_STEP	5127	Impossibile passare alla successiva voce di richiesta
ERR_DATABASE_NOT_READY	5128	I dati per la lettura dei risultati della richiesta non sono ancora pronti
ERR_DATABASE_BIND_PARAMETERS	5129	Impossibile sostituire automaticamente i parametri con una richiesta SQL
Operazioni con WebRequest		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	URL non valido
ERR_WEBREQUEST_CONNECT_FAILED	5201	Impossibile connettersi all' URL specificato
ERR_WEBREQUEST_TIMEOUT	5202	È stato superato il timeout
ERR_WEBREQUEST_REQUEST_FAILED	5203	La richiesta HTTP non è riuscita

Costante	Codice	Descrizione
Operazioni con la rete (sockets)		
ERR_NETSOCKET_INVALIDHANDLE	5270	L'handle del socket non valido è passato alla funzione
ERR_NETSOCKET_TOO_MANY_OPENED	5271	Troppi socket aperti (max 128)
ERR_NETSOCKET_CANNOT_CONNECT	5272	Impossibile connettersi all'host remoto
ERR_NETSOCKET_IO_ERROR	5273	Impossibile inviare/ricevere dati dal socket
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	Impossibile stabilire una connessione sicura (TLS Handshake)
ERR_NETSOCKET_NO_CERTIFICATE	5275	Nessun dato sul certificato che protegge la connessione
Simboli personalizzati		
ERR_NOT_CUSTOM_SYMBOL	5300	È necessario specificare un simbolo personalizzato
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	Il nome del simbolo personalizzato non è valido. Il nome del simbolo può contenere solo lettere latine senza punteggiatura, spazi o caratteri speciali (può contenere solo ".", "_", "&" e "#"). Non è consigliabile utilizzare i caratteri <, >, :, ", /,\, |, ?, *.
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	Il nome del simbolo personalizzato è troppo lungo. La lunghezza del nome del simbolo non deve superare i 32 caratteri tra cui il carattere finale 0
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	Il percorso del simbolo personalizzato è troppo lungo. La lunghezza del percorso non deve superare i 128 caratteri tra cui "Custom\\", il nome del simbolo, i separatori di gruppo e la fine 0
ERR_CUSTOM_SYMBOL_EXIST	5304	Esiste già un simbolo personalizzato con lo stesso nome
ERR_CUSTOM_SYMBOL_ERROR	5305	Errore durante la creazione, la cancellazione o la modifica del simbolo personalizzato

Costante	Codice	Descrizione
ERR_CUSTOM_SYMBOL_SELECTED	5306	Stai cercando di eliminare un simbolo personalizzato selezionato in Market Watch
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	Una proprietà di simboli personalizzati non valida
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	Un parametro sbagliato durante l'impostazione della proprietà di un simbolo personalizzato
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	Un parametro di stringa troppo lungo impostando la proprietà di un simbolo personalizzato
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	I ticks nell'array sono non organizzati nell'ordine di tempo
Calendario Economico		
ERR_CALENDAR_MORE_DATA	5400	La grandezza dell'array è insufficiente per la ricezione delle descrizioni di tutti i valori
ERR_CALENDAR_TIMEOUT	5401	Limite di tempo richiesto superato
ERR_CALENDAR_NO_DATA	5402	Paese non trovato
Lavorare con i database		
ERR_DATABASE_ERROR	5601	Errore generico
ERR_DATABASE_LOGIC	5602	Errore logico interno di SQLite
ERR_DATABASE_PERM	5603	Accesso negato
ERR_DATABASE_ABORT	5604	La routine di callback ha richiesto l'interruzione
ERR_DATABASE_BUSY	5605	File di database bloccato
ERR_DATABASE_LOCKED	5606	Tabella del database bloccata
ERR_DATABASE_NOMEM	5607	Memoria insufficiente per il completamento dell'operazione
ERR_DATABASE_READONLY	5608	Tentativo di scrittura nel database di sola lettura
ERR_DATABASE_INTERRUPT	5609	Operazione terminata da sqlite3_interrupt()
ERR_DATABASE_IOERR	5610	Errore I/O del disco
ERR_DATABASE_CORRUPT	5611	Immagine del disco del database danneggiata

Costante	Codice	Descrizione
ERR_DATABASE_NOTFOUND	5612	Codice operazione sconosciuto in sqlite3_file_control()
ERR_DATABASE_FULL	5613	Inserimento non riuscito perché il database è pieno
ERR_DATABASE_CANTOPEN	5614	Impossibile aprire il file del database
ERR_DATABASE_PROTOCOL	5615	Errore di blocco del protocollo del database
ERR_DATABASE_EMPTY	5616	Solo per uso interno
ERR_DATABASE_SCHEMA	5617	Schema del database modificato
ERR_DATABASE_TOOBIG	5618	String o BLOB supera il limite di dimensioni
ERR_DATABASE_CONSTRAINT	5619	Annullato a causa di violazione del vincolo
ERR_DATABASE_MISMATCH	5620	Mancata corrispondenza del tipo di dati
ERR_DATABASE_MISUSE	5621	Libreria utilizzata in modo errato
ERR_DATABASE_NOLFS	5622	Utilizzo funzionalità del sistema operativo non supportate sull'host
ERR_DATABASE_AUTH	5623	Autorizzazione negata
ERR_DATABASE_FORMAT	5624	Non usato
ERR_DATABASE_RANGE	5625	Errore parametro bind, indice errato
ERR_DATABASE_NOTADB	5626	File aperto che non è un file di database
Metodi Matriciali e Vettoriali		
ERR_MATRIX_INTERNAL	5700	Errore interno del sottosistema di esecuzione della matrice/vettore
ERR_MATRIX_NOT_INITIALIZED	5701	Matrice/vettore non inizializzato
ERR_MATRIX_INCONSISTENT	5702	Dimensioni incoerenti delle matrici/vettori nell'operazione
ERR_MATRIX_INVALID_SIZE	5703	Dimensione matrice/vettore non valida
ERR_MATRIX_INVALID_TYPE	5704	Tipo di matrice/vettore non valido

Costante	Codice	Descrizione
ERR_MATRIX_FUNC_NOT_ALLOWED	5705	Funzione non disponibile per questa matrice/vettore
ERR_MATRIX_CONTAINS_NAN	5706	Matrice/vettore contiene non numeri (Nan/Inf)
Modelli ONNX		
ERR_ONNX_INTERNAL	5800	Errore interno ONNX
ERR_ONNX_NOT_INITIALIZED	5801	Errore di inizializzazione ONNX Runtime API
ERR_ONNX_NOT_SUPPORTED	5802	Proprietà o valore non supportati da MQL5
ERR_ONNX_RUN_FAILED	5803	Errore di esecuzione ONNX runtime API
ERR_ONNX_INVALID_PARAMETERS_COUNT	5804	Numero di parametri non validi passati a OnnxRun
ERR_ONNX_INVALID_PARAMETER	5805	Valore del parametro non valido
ERR_ONNX_INVALID_PARAMETER_TYPE	5806	Tipo di parametro non valido
ERR_ONNX_INVALID_PARAMETER_SIZE	5807	Dimensione del parametro non valida
ERR_ONNX_WRONG_DIMENSION	5808	Dimensione tensore non impostata o non valida
Errori Definiti-Dall-utente		
ERR_USER_ERROR_FIRST	65536	Errori definiti dall'utente iniziano con questo codice

Vedi anche

Codici di Ritorno del Trade Server

Costanti Input ed Output

Costanti:

- Flags aperturare file
- Proprietà file
- Posizionamento all'interno di un file
- Utilizzo codice pagina
- MessageBox

Flags di Apertura File

Valori Flag di apertura file specificano la modalità di accesso ai file. I Flags sono definiti come segue:

Identificatore	Valore	Descrizione
FILE_READ	1	Il file viene aperto per la lettura. Flag viene usato in FileOpen(). Quando si apre un file, la specifica di file FILE_WRITE e/o FILE_READ è necessaria.
FILE_WRITE	2	Il file è aperto in scrittura. Flag viene usato in FileOpen(). Quando si apre un file, la specifica di file FILE_WRITE e/o FILE_READ è necessaria.
FILE_BIN	4	Modalità binaria lettura/scrittura (senza conversione stringa a stringa). Flag viene usato in FileOpen()
FILE_CSV	8	CSV file (tutti i suoi elementi vengono convertiti in stringhe di tipo appropriato, unicode o ANSI, e separati da separatore). Flag viene usato in FileOpen()
FILE_TXT	16	File di testo semplice (lo stesso di file csv, ma senza prendere in considerazione i separatori). Flag viene usato in FileOpen()
FILE_ANSI	32	Stringhe di tipo ANSI (uno dei simboli byte). Flag viene usato in FileOpen()
FILE_UNICODE	64	Stringhe di tipo UNICODE (due simboli di byte). Flag viene usato in FileOpen()
FILE_SHARE_READ	128	L'accesso condiviso per la lettura da diversi programmi. Il flag viene usato in FileOpen(), ma non sostituisce la necessità di indicare FILE_WRITE e/o il flag FILE_READ all'apertura di un file.
FILE_SHARE_WRITE	256	L'accesso condiviso per la scrittura da diversi programmi. Il flag viene usato in FileOpen(), ma non sostituisce la necessità di indicare FILE_WRITE e/o il flag FILE_READ all'apertura di un file.
FILE_REWRITE	512	Possibilità, per la riscrittura del file utilizzando le funzioni di FileCopy() e FileMove(). Il file deve esistere o deve essere aperto in scrittura, altrimenti il file non viene aperto.
FILE_COMMON	4096	Il percorso del file nella cartella comune di tutti i terminali client \Terminal\Common\Files. Il flag viene usato nelle funzioni FileOpen(), FileCopy(), FileMove() e FileIsExist().

Una o più flags possono essere specificato quando si apre un file. Si tratta di una combinazione di flag. La combinazione di flag viene scritta utilizzando il segno logico OR (|), che è posizionato tra flags enumerati. Ad esempio, per aprire un file in formato CSV per la lettura e la scrittura, allo stesso tempo, specificare la combinazione FILE_READ|FILE_WRITE|FILE_CSV.

Esempio:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Ci sono alcune caratteristiche specifiche di lavoro quando si specificano flags di lettura e scrittura:

- Se viene specificato FILE_READ, viene effettuato un tentativo di aprire un file esistente. Se il file non esiste, l'apertura del file non riesce, un nuovo file non viene creato.
- FILE_READ|FILE_WRITE – un nuovo file viene creato se il file con il nome specificato non esiste.
- FILE_WRITE – il file viene creato di nuovo con una dimensione pari a zero.

Quando si apre un file, è necessaria la specifica di FILE_WRITE e/o FILE_READ.

Flags che definiscono il tipo di lettura di un file aperto, posseggono priorità. La più alta flag è FILE_CSV, poi va FILE_BIN, e FILE_TXT è di priorità più bassa. Quindi, se le bandiere vengono indicate al tempo stesso, (FILE_TXT | FILE_CSV o FILE_TXT | FILE_BIN o FILE_BIN | FILE_CSV), verrà utilizzata la bandiera con la priorità più alta.

Bandiere che definiscono il tipo di codifica anche hanno la priorità. FILE_UNICODE sono di una priorità maggiore rispetto FILE_ANSI. Quindi, se si specifica la combinazione FILE_UNICODE|FILE_ANSI, verrà utilizzato il flag FILE_UNICODE.

Se né FILE_UNICODE né FILE_ANSI sono indicati, FILE_UNICODE è implicito. Se non FILE_CSV, né FILE_BIN, né FILE_TXT sono specificati, FILE_CSV è implicito.

Se un file viene aperto per la lettura come file di testo (o FILE_TXT FILE_CSV), ed all'inizio del file viene trovata di una speciale indicazione due-byte 0xff,0xfe, il flag di codifica sarà FILE_UNICODE, anche se viene specificato FILE_ANSI.

Vedi anche

Funzioni con i File

Proprietà file

La funzione FileGetInteger() viene utilizzata per ottenere le proprietà del file. L'identificatore della proprietà richiesta dell'enumerazione ENUM_FILE_PROPERTY_INTEGER viene passato ad esso durante la chiamata.

ENUM_FILE_PROPERTY_INTEGER

ID	ID descrizione
FILE_EXISTS	Controllare l'esistenza
FILE_CREATE_DATE	Data di creazione
FILE_MODIFY_DATE	Data dell'ultima modifica
FILE_ACCESS_DATE	Data dell'ultimo accesso al file
FILE_SIZE	Dimensione del file in byte
FILE_POSITION	Posizione di un puntatore nel file
FILE_END	Prende la fine del segno del file
FILE_LINE_END	Prendi la fine del segno linea
FILE_IS_COMMON	Il file viene aperto in una cartella condivisa di tutti i terminali (vedi FILE_COMMON)
FILE_IS_TEXT	Il file viene aperto come file di testo (vedi FILE_TXT)
FILE_IS_BINARY	Il file viene aperto come un file binario (vedi FILE_BIN)
FILE_IS_CSV	Il file viene aperto come CSV (vedi FILE_CSV)
FILE_IS_ANSI	Il file viene aperto come ANSI (vediFILE_ANSI)
FILE_IS_READABLE	Il file aperto è leggibile (vedi File_read)
FILE_IS_WRITABLE	Il file aperto è scrivibile (vedi File_write)

La funzione FileGetInteger() ha due diverse opzioni di chiamata. Nella prima opzione, per ottenere le proprietà di un file, viene specificato il suo handle, che viene ottenuto durante l'apertura del file utilizzando la funzione FileOpen(). Questa opzione permette di ottenere tutte le proprietà di un file.

La seconda opzione della funzione FileGetInteger() restituisce i valori delle proprietà del file, per il nome del file. Utilizzando questa opzione, solo le seguenti proprietà generali possono essere ottenute:

- FILE_EXISTS - esistenza di un file con un nome specificato
- FILE_CREATE_DATE - data di creazione del file con il nome specificato
- FILE_MODIFY_DATE - data di modifica del file con il nome specificato
- FILE_ACCESS_DATE - data dell'ultimo accesso al file con il nome specificato
- FILE_SIZE - la dimensione del file con il nome specificato

Quando si cerca di ottenere le proprietà diverse da quelle di cui sopra, la seconda opzione di FileGetInteger() chiamata restituirà un errore.

Posizionamento all'interno di un file

La maggior parte delle funzioni di file sono associate ad operazioni di lettura/scrittura dati. Allo stesso tempo, mediante l' FileSeek() è possibile specificare la posizione di un puntatore di file in una posizione all'interno del file, da cui l'operazione di lettura o scrittura sarà eseguita. L'enumerazione ENUM_FILE_POSITION contiene valide posizioni puntatore, rispetto al quale è possibile specificare lo spostamento in byte per l'operazione successiva.

ENUM_FILE_POSITION

Identificatore	Descrizione
SEEK_SET	Inizio file
SEEK_CUR	Posizione attuale di un puntatore a file
SEEK_END	Fine file

Vedi anche

FileIsEnding, FileIsLineEnding

Utilizzo di una Tabella Codici in Operazioni di Conversione di Stringhe

Quando si convertono variabili stringa in array di tipo char e precedenti, utilizzato in MQL5 la codifica di default corrisponde all' attuale ANSI del sistema operativo Windows (CP_ACP). Se si desidera specificare un diverso tipo di codifica, può essere impostato come parametro aggiuntivo per le funzioni CharArrayToString(), StringToCharArray() e FileOpen().

La tabella elenca le costanti predefinite per alcune delle più popolari pagine di codice. Pagine di codice non menzionate possono essere specificate da un codice corrispondente alla pagina.

Costanti built-in di Tabelle Codici

Constant	Valore	Descrizione
CP_ACP	0	L'attuale codepage Windows ANSI
CP_OEMCP	1	L'attuale sistema OEM code page.
CP_MACCP	2	L'attuale code page di sistema Macintosh. **Nota:** Questo valore è usato soprattutto nei codici di programma creati prima, e non serve a niente ora, dal momento che i moderni computer Macintosh usano Unicode per la codifica.
CP_THREAD_ACP	3	La tabella codici ANSI di Windows per il thread corrente.
CP_SYMBOL	42	Simbolo codice della pagina
CP_UTF7	65000	UTF-7 code page.
CP_UTF8	65001	UTF-8 code page.

Vedi anche

Proprietà del Terminale Client

Costanti della finestra MessageBox

Questa sezione contiene i codici di ritorno della funzione MessageBox(). Se una finestra di messaggio contiene un pulsante Annulla, la funzione restituisce IDCANCEL, nel caso in cui il tasto ESC o il pulsante Annulla viene premuto. Se non c'è il pulsante Annulla nella finestra del messaggio, la pressione di ESC non dà alcun effetto.

Constant	Valore	Descrizione
IDOK	1	Bottone "OK" è stato premuto
IDCANCEL	2	Bottone "Cancel" è stato premuto
IDABORT	3	Bottone "Interruzione" è stato premuto
IDRETRY	4	Bottone "Riprova" è stato premuto
IDIGNORE	5	Bottone "Ignora" è stato premuto
IDYES	6	Bottone "Sì" è stato premuto
IDNO	7	Bottone "No" è stato premuto
IDTRYAGAIN	10	Bottone "Prova Ancora" è stato premuto
IDCONTINUE	11	Bottone "Continua" è stato premuto

Le flags principali della funzione MessageBox() definiscono il contenuto ed il comportamento della finestra di dialogo. Questo valore può essere una combinazione dei seguenti gruppi flags:

Constant	Valore	Descrizione
MB_OK	0x00000000	La finestra di messaggio contiene un solo bottone: OK. Default
MB_OKCANCEL	0x00000001	Finestra di messaggio contiene due pulsanti: OK e Annulla
MB_ABORTRETRYIGNORE	0x00000002	Finestra di messaggio contiene tre pulsanti: Interrompi, Riprova ed Ignora
MB_YESNOCANCEL	0x00000003	Finestra di messaggio contiene tre pulsanti: Sì, No e Annulla
MB_YESNO	0x00000004	Finestra di messaggio contiene due pulsanti: Sì e No
MB_RETRYCANCEL	0x00000005	Finestra di messaggio contiene due pulsanti: Riprova e Annulla
MB_CANCELTRYCONTINUE	0x00000006	Finestra di messaggio contiene tre pulsanti: Annulla, Riprova, Continua

Per visualizzare l'icona nella finestra del messaggio, è necessario specificare flag addizionali:

Constant	Valore	Descrizione
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	L'icona del segno di STOP
MB_ICONQUESTION	0x00000020	L'icona del segno di punto interrogativo
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	L'icona del segno di esclamazione/avvertimento
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	Il segno i cerchiata

Bottoni di default vengono definiti dalle seguenti flag:

Constant	Valore	Descrizione
MB_DEFBUTTON1	0x00000000	Il primo bottone MB_DEFBUTTON1 - è di default, se gli altri pulsanti MB_DEFBUTTON2, MB_DEFBUTTON3 o MB_DEFBUTTON4 non sono specificati
MB_DEFBUTTON2	0x00000100	Il secondo bottone è default
MB_DEFBUTTON3	0x00000200	Il terzo bottone è default
MB_DEFBUTTON4	0x00000300	Il quarto bottone è default

MQL5 Programs

Per far opeare il programma mql5, esso deve essere compilato (bottone Compile o tasto F7). La compilazione dovrebbe passare senza errori (sono possibili alcuni avvisi, ma devono essere analizzati). In questo processo, un file eseguibile con lo stesso nome e con estensione EX5 dev' essere creato nella directory corrispondente, terminal_dir\MQL5\Experts, terminal_dir\MQL5\indicators or terminal_dir\MQL5\scripts. Questo file può essere eseguito.

Le caratteristiche operative dei programmi MQL5 sono descritte nelle seguenti sezioni:

- Programma in corso - ordine di chiamata event-handler predefiniti.
- Testing di Strategie di Trading - caratteristiche di funzionamento programmi MQL5 nello Strategy Tester.
- Eventi terminale client - descrizione degli eventi, che possono essere elaborati nei programmi.
- Chiamata delle funzioni importate - descrizione ordine, parametri consentiti, i dettagli di ricerca ed accordi di chiamata per le funzioni importate.
- Errori di runtime - ottenere informazioni sugli errori di runtime e critici.

Expert Advisor, indicatori personalizzati e script sono collegati ad uno dei grafici aperti con il metodo Drag'n'Drop dalla finestra del Navigator.

Affinché un Expert Advisor smetta di funzionare, dovrebbe essere rimosso dal grafico. Per farlo selezionare "Lista Experts" nel menu contestuale del grafico, quindi selezionare un Expert Advisor dalla lista e fare clic sul pulsante "Rimuovi". Il funzionamento degli Expert Advisors è influenzato anche dallo stato del bottone "AutoTrading".

Per fermare un indicatore personalizzato, esso deve essere rimosso dal grafico.

Gli indicatori personalizzati ed Expert Advisors lavorano fino a quando non vengono esplicitamente rimossi da un grafico, le informazioni su Expert Advisor ed Indicatori attaccati vengono salvate tra le sessioni di terminale client.

Gli script vengono eseguiti una sola volta e vengono eliminati automaticamente al termine del completamento dell'operazione o al cambio dello stato del grafico corrente, o su arresto del terminale client. Dopo il riavvio del terminale client, gli scripts non vengono avviati, perché le informazioni su di essi non vengono salvate.

Massimo un Expert Advisor, uno script ed un numero illimitato di indicatori, possono operare in un grafico.

I servizi non richiedono di essere associati ad un chart per funzionare e sono progettati per eseguire funzioni ausiliarie. Ad esempio, in un servizio, è possibile creare un simbolo personalizzato, aprire il suo chart, ricevere i dati per esso in un ciclo infinito usando le funzioni di rete e aggiornarlo costantemente.

I programmi in esecuzione

Ogni script ed ogni Expert Advisor viene eseguito in un thread separato. Tutti gli indicatori calcolati su un simbolo, anche se sono collegati ai grafici diversi, lavorano nello stesso thread. Dunque, tutti gli indicatori di uno simbolo condividono le risorse di un thread.

Tutte le altre azioni associate ad un simbolo, come l'elaborazione di ticks e la sincronizzazione dello storico, sono sempre eseguiti nello stesso thread con gli indicatori. Ciò significa che se un' azione infinita viene eseguita in un indicatore, tutti gli altri eventi associati con il suo simbolo non verranno mai eseguiti.

Durante l'esecuzione di un Expert Advisor, assicurarsi che ha un vero e proprio ambiente di trading e può accedere allo storico del simbolo e periodo richiesti, e sincronizzare i dati tra il terminale ed il server. Per tutte queste procedure, il terminale fornisce un ritardo di avvio di non più di 5 secondi, dopo di che l'Expert Advisor viene avviato con i dati disponibili. Pertanto, nel caso in cui non vi è alcun collegamento al server, questo può portare ad un ritardo nella partenza di un Expert Advisor.

La tabella che segue contiene un breve sommario di programmi mql5:

Programma	Funzionamento	Nota
Servizio	Un thread separato, il numero di thread per i servizi è uguale al numero di servizi	Un servizio in loop non può interrompere l'esecuzione di altri programmi
Script	Un thread separato, il numero di threads per script è uguale al numero di script	Uno script in loop non può interrompere l'esecuzione di altri programmi
Expert Advisor	Un thread separato, il numero di thread di Expert Advisors è uguale al numero di Expert Advisors	An Expert Advisor in loop non può interrompere l'esecuzione di altri programmi
Indicatore	Un thread per tutti gli indicatori su un simbolo. Il numero di thread è uguale al numero di simboli con indicatori	Un loop infinito in un indicatore cesserà tutti gli altri indicatori di questo simbolo

Subito dopo che un programma è collegato ad un grafico, viene caricato nella memoria del terminale client, così come le variabili globali vengono inizializzate. Se qualche variabile globale di tipo classe ha un costruttore, questo costruttore sarà chiamato durante l'inizializzazione di variabili globali.

Dopo di che il programma resta in attesa di un evento dal terminale client. Ogni programma-mql5 deve avere almeno un event-handler, altrimenti il programma caricato non verrà eseguito. Gli event handlers hanno nome, parametri e tipi restituiti, tutti predefiniti.

Tipo	Nome funzione	Parametri	Applicazione	Comment
int	OnInit	none	Expert Advisors ed indicatori	Init event handler. Permette di utilizzare il tipo di ritorno void.

Tipo	Nome funzione	Parametri	Applicazione	Comment
void	OnDeinit	const int reason	Expert Advisors ed indicatori	Deinit event handler.
void	OnStart	none	Script e Servizi	Start event handler.
int	OnCalculate	const int rates_total, const int prev_calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume[], const long &Volume[], const int &Spread[]	indicators	Calcola gli event handlers per tutti i prezzi.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	indicators	Calculate event handler su un singolo array di dati. L'indicatore non può avere due event handlers simultaneamente. In questo caso l'unico event handler lavorerà sull' array di dati.
void	OnTick	none	Expert Advisors	NewTick event handler. Mentre viene elaborato l'event della ricevuta di un nuovo tick, nessun altro evento di questo tipo viene ricevuto.
void	OnTimer	none	Expert Advisors ed indicatori	Timer event handler.
void	OnTrade	none	Expert Advisors	Trade event handler.
double	OnTester	none	Expert Advisors	Tester event handler.
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	Expert Advisors ed indicatori	ChartEvent event handler.

Tipo	Nome funzione	Parametri	Applicazione	Comment
void	OnBookEvent	const string &symbol_name	Expert Advisors ed indicatori	BookEvent event handler.

Un terminale client invia nuovi eventi ai corrispondenti grafici aperti. Gli eventi possono essere generati anche da grafici (grafico eventi) o programmi-mql5 (eventi personalizzati). Generazione di eventi di creazione o l'eliminazione di oggetti grafici in un grafico possono essere abilitati o disabilitati impostando le proprietà del grafico CHART_EVENT_OBJECT_CREATE e CHART_EVENT_OBJECT_DELETE. Ogni programma mql5 ed ogni grafico ha la propria coda di eventi, in cui vengono aggiunti tutti i nuovi eventi in arrivo.

Un programma riceve solo gli eventi dal grafico su cui gira. Tutti gli eventi vengono elaborati uno dopo l'altro nell'ordine in cui vengono ricevuti. Se una coda ha già un evento NewTick, o questo evento è attualmente in fase di elaborazione, il nuovo evento NewTick non viene inserito nella coda del programma mql5. Analogamente, se ChartEvent è già accodato, o questo evento è in fase di elaborazione, nessun nuovo evento di questo tipo viene messo in fila. Gli eventi timer vengono gestiti allo stesso modo - se l'evento Timer è in coda o manipolato, l'evento nuovo timer non è messo in coda.

Code degli eventi hanno una grandezza limitata ma sufficiente, in modo che l'overflow della coda per programmi ben scritti è improbabile. In caso di overflow della coda, i nuovi eventi vengono ignorati senza fare la fila.

Si raccomanda vivamente di non utilizzare loop infiniti per gestire gli eventi. Le possibili eccezioni sono script e servizi che gestiscono un singolo evento Start.

Le Librerie non gesticono nessuno degli eventi.

Funzioni vietate in Indicatori ed Expert Advisor

Indicatori, script ed Expert Advisor sono programmi eseguibili scritti in MQL5. Essi sono progettati per diversi tipi di compiti. Pertanto vi sono alcune limitazioni all'uso di determinate funzioni, a seconda del tipo di programma. Le seguenti funzioni sono vietate negli indicatori:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();
- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

Tutte le funzioni progettate per gli indicatori sono vietate in Expert Advisor e script:

- SetIndexBuffer();
- IndicatorSetDouble();

- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

La libreria non è un programma indipendente ed è eseguita nel contesto del programma MQL5 che lo ha chiamato: script, indicatore o Expert Advisor. Di conseguenza, le restrizioni di cui sopra si applicano alla libreria chiamata.

Funzioni vietate nei servizi

I servizi non accettano alcun evento, in quanto non sono associati ad un chart. Le seguenti funzioni sono proibite nei servizi:

ExpertRemove();

EventSetMillisecondTimer();

EventSetTimer();

EventKillTimer();

SetIndexBuffer();

IndicatorSetDouble();

IndicatorSetInteger();

IndicatorSetString();

PlotIndexSetDouble();

PlotIndexSetInteger();

PlotIndexSetString();

PlotIndexGetInteger();

Caricamento e decaricamento di Indicatori

Gli indicators vengono caricati nei casi seguenti:

- un indicatore è attaccato ad un grafico;
- avvio del terminale (se l'indicatore viene attaccato al grafico prima della chiusura del terminale);
- caricamento di un template (se l'indicatore attaccato ad un grafico viene specificato nel template);
- modifica di un profilo (se l'indicatore è attaccato ad uno dei grafici del profilo);
- modifica di un simbolo e/o tempi timeframe di un grafico, al quale l'indicatore è attaccato;
- modifica dell'account sul quale il terminale è collegato;
- dopo la ricompilazione con successo di un indicatore, se l'indicatore è stato attaccato ad un grafico;
- cambiamento di parametri di input dell'indicatore.

Gli indicatori sono decaricati nei seguenti casi:

- quando si stacca un indicatore da un grafico;
- arresto terminale (se l'indicatore è stato attaccato ad un grafico);
- caricamento di un template, se un indicatore è attaccato ad un grafico;
- chiusura di un grafico, a cui l'indicatore è attaccato;
- modifica di un profilo, se l'indicatore è attaccato a uno dei grafici del profilo modificato;
- modifica di un simbolo e/o tempi timeframe di un grafico, al quale l'indicatore è attaccato;
- modifica dell'account sul quale il terminale è collegato;
- cambiamento di parametri di input dell'indicatore.

Caricamento e decaricamento di Expert Advisors

Expert Advisors vengono caricati nei seguenti casi:

- quando si attacca un Expert Advisor ad un grafico;
- avvio del terminale (se l' Expert Advisor viene attaccato al grafico prima della chiusura del terminale);
- caricamento di un template (se l'Expert Advisor attaccato ad un grafico viene specificato nel template);
- modifica di un profilo (se l'Expert Advisor è attaccato ad uno dei grafici del profilo);
- connessione ad un account, anche se il numero di conto è lo stesso (se l'Expert Advisor è stata attaccato al grafico prima dell'autorizzazione del terminale sul server).

Expert Advisors vengono decaricati nei seguenti casi:

- quando si stacca un Expert Advisor da un grafico;
- se un nuovo Expert Advisor è collegato ad un grafico, se un altro Expert Advisor è stato già attaccato, questo Expert Advisor viene decaricato.
- arresto terminale (se l'Expert Advisor è stato attaccato ad un grafico);
- caricamento di un template, se un Expert Advisor è attaccato ad un grafico;
- chiusura di un grafico, a cui l' Expert Advisor è attaccato;
- modifica di un profilo, se l' Expert Advisor è attaccato a uno dei grafici del profilo modificato;
- modifica dell'account sul quale il terminale è collegato (se l'Expert Advisor è stato attaccato al grafico prima dell'autorizzazione del terminale sul server;
- calling the ExpertRemove() function.

Nel caso in cui il simbolo o il timeframe di un grafico, al quale è attaccato l'Expert Advisor, cambia, gli Expert Advisor non vengono caricati o decaricati. In questo caso il terminale client chiama in subsequenzialmente i gestori OnDeinit() sul vecchio simbolo/timeframe ed OnInit() sul nuovo simbolo/timeframe (se ce ne sono), i valori delle variabili globali e variabili statiche non vengono ripristinati. Tutti gli eventi, che sono stati ricevuti per l'Expert Advisor prima che l'inizializzazione sia stata completata (funzione OnInit()) vengono saltati.

Caricamento e decaricamento di Scripts

Gli script vengono caricati immediatamente dopo che vengono attaccati ad un grafico e decaricati immediatamente dopo aver completato il loro funzionamento. OnInit () e OnDeinit () non vengono chiamati per gli script.

Quando un programma viene scaricato (eliminato da un grafico) il terminale client esegue la deinizializzazione delle variabili globali ed elimina la coda di eventi. In questo caso la deinizializzazione significa azzeramento di tutte le variabili di tipo string-, deallocazione dioggetti array dinamici e la chiamata dei loro distruttori se sono disponibili.

Caricamento e decaricamento dei Servizi

I servizi vengono caricati subito dopo l'avvio del terminale se sono stati avviati al momento dello spegnimento del terminale. I servizi vengono scaricati immediatamente dopo aver completato il loro lavoro.

I servizi hanno un singolo gestore OnStart(), in cui è possibile implementare un ciclo infinito di ricezione e gestione dei dati, ad esempio creando e aggiornando simboli personalizzati utilizzando le funzioni di rete.

A differenza di Expert Advisors, indicatori e script, i servizi non sono associati ad un chart specifico, pertanto viene fornito un meccanismo separato per avviarli. Una nuova istanza di servizio viene creata nel Navigator utilizzando il comando "Aggiungi servizio". Un'istanza di servizio può essere avviata, arrestata e rimossa utilizzando il menu di istanza appropriato. Per gestire tutte le istanze, utilizzare il menu di servizio.

Per una migliore comprensione del funzionamento degli Expert Advisor si consiglia di compilare il codice del seguente Expert Advisor ed eseguire operazioni di carico/decarico, cambio template, cambio simbolo, cambio timeframe ecc.:

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            TestExpert.mq5 |
//|                                  Copyright 2009, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
  {
public:
   CTestClass() { Print("CTestClass constructor"); }
   ~CTestClass() { Print("CTestClass destructor"); }
  };
```

```
CTestClass global;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                        |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Inizializzazione");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Deinizializzazione con motivo",reason);
  }
//+------------------------------------------------------------------+
//| Funzione tick dell'Expert                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

Vedi anche

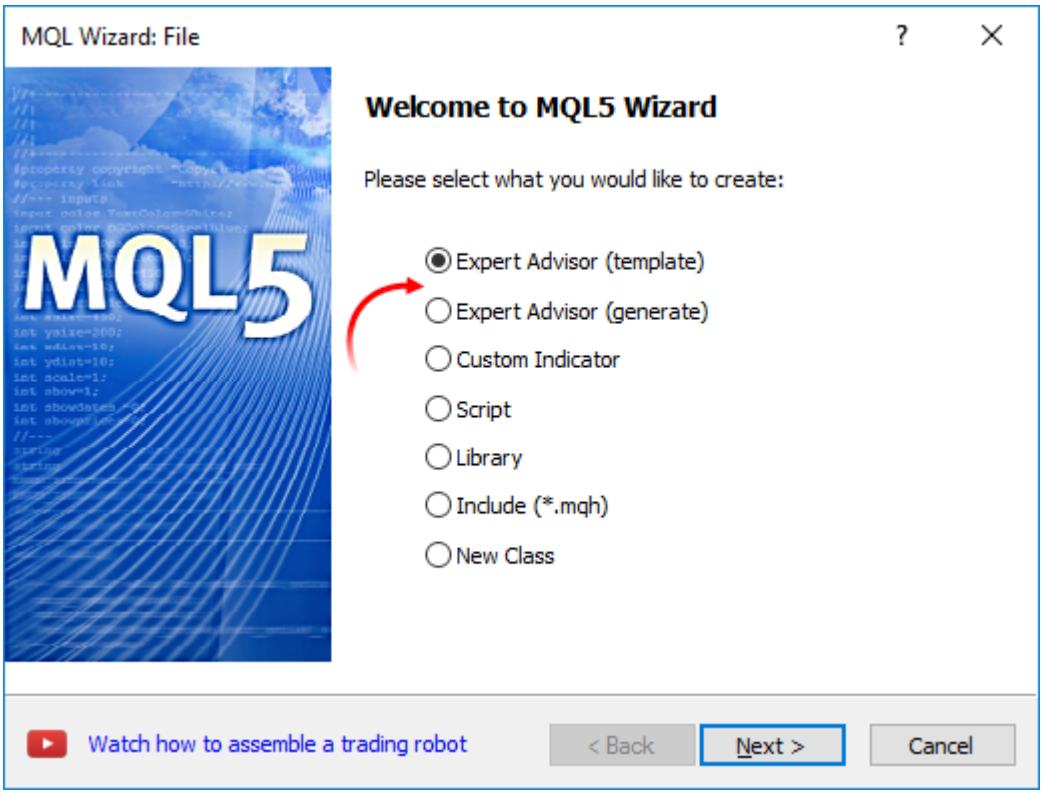
Eventi terminale client, Event handlers

Trade Permission

Trade Automation

MQL5 language provides a special group of trade functions designed for developing automated trading systems. Programs developed for automated trading with no human intervention are called Expert Advisors or trading robots. In order to create an Expert Advisor in MetaEditor, launch MQL5 Wizard and select one of the two options:

- Expert Advisor (template) - allows you to create a template with ready-made event handling functions that should be supplemented with all necessary functionality by means of programming.
- Expert Advisor (generate) - allows you to develop a full-fledged trading robot simply by selecting the necessary modules: trading signals module, money management module and trailing stop module.



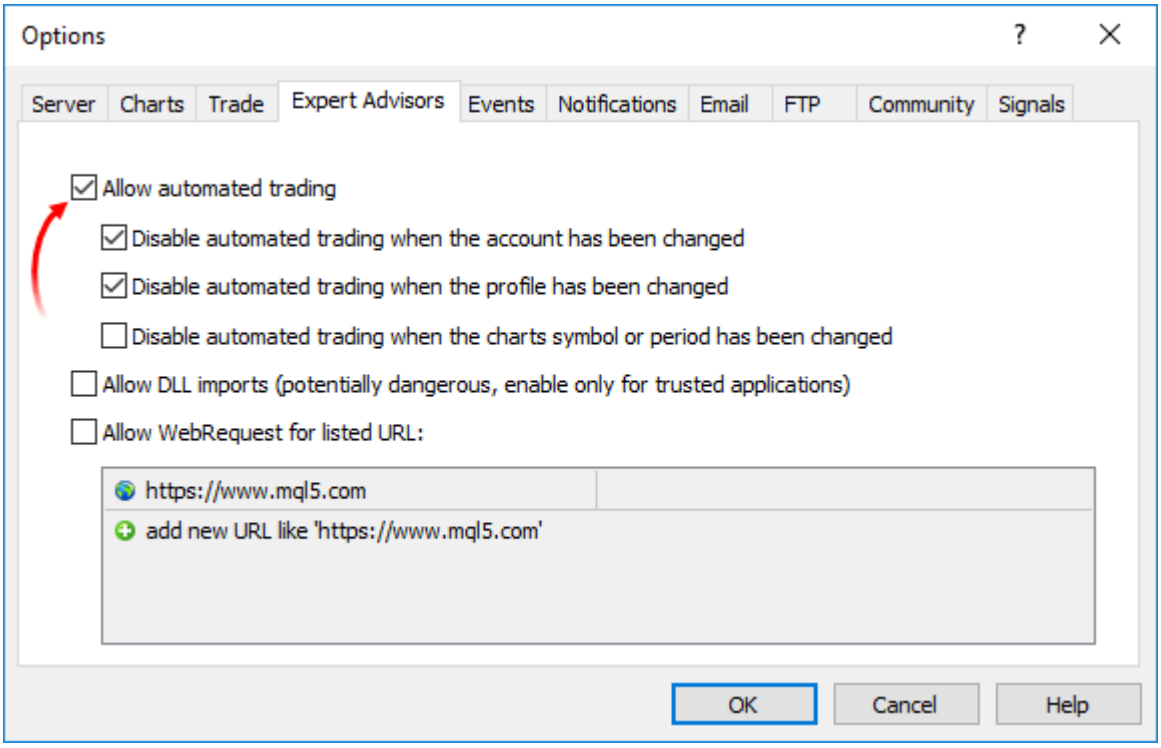
Trading functions can work only in Expert Advisors and scripts. Trading is not allowed for indicators.

Checking for Permission to Perform Automated Trading

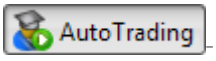
In order to develop a reliable Expert Advisor capable of working without human intervention, it is necessary to arrange a set of important checks. First, we should programmatically check if trading is allowed at all. This is a basic check that is indispensable when developing any automated system.

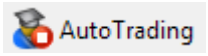
Checking for permission to perform automated trading in the terminal

The terminal settings provide you with an ability to allow or forbid automated trading for all programs.



You can switch automated trading option right on the terminal's Standard panel:

- 

- automated trading enabled, trading functions in launched applications are allowed for use.
- 

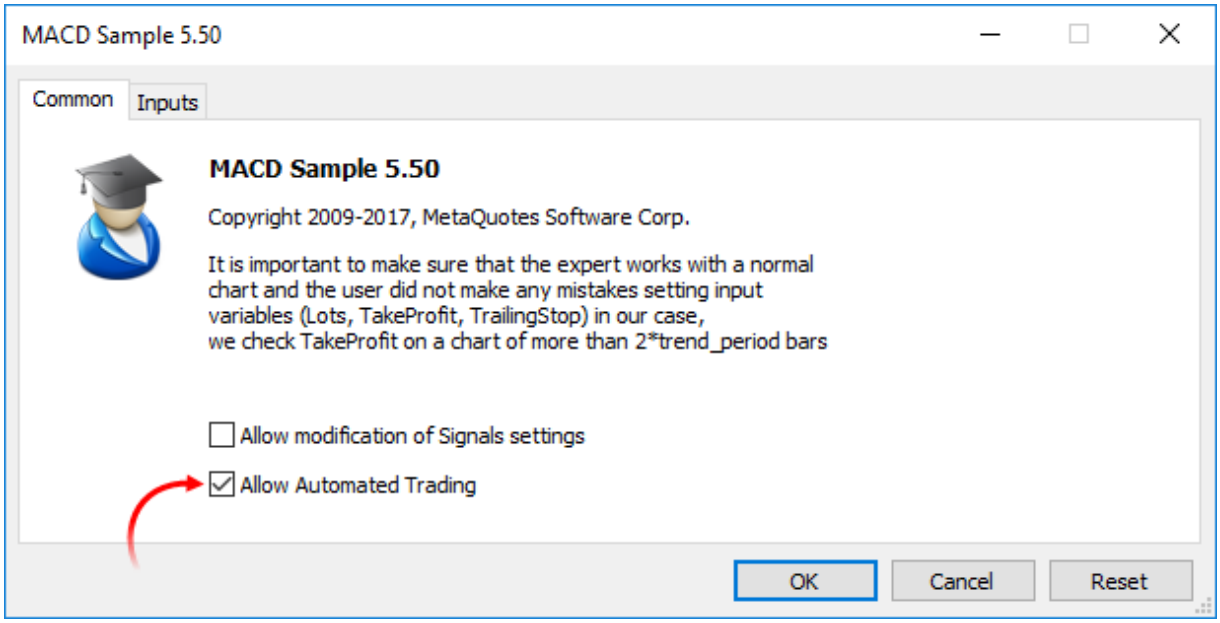
- automated trading disabled, running applications are unable to execute trading functions.

Sample check:

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Check if automated trading is allowed in the terminal settings!");
```

Checking if trading is allowed for a certain running Expert Advisor/script

You can allow or forbid automated trading for a certain program when launching it. To do this, use the special check box in the program properties.



Sample check:

```
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
      Alert("Check if automated trading is allowed in the terminal settings!");
   else
     {
      if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
         Alert("Automated trading is forbidden in the program settings for ",__FILE__)
     }
```

Checking if trading is allowed for any Expert Advisors/scripts for the current account

Automated trading can be disabled at the trade server side. Sample check:

```
   if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
      Alert("Automated trading is forbidden for the account ",AccountInfoInteger(ACCOU
      " at the trade server side");
```

If automated trading is disabled for a trading account, trading operations of Expert Advisors/scripts are not executed.

Checking if trading is allowed for the current account

In some cases, any trading operations are disabled for a certain trading account - neither manual nor automated trading can be performed. Sample check when an investor password has been used to connect to a trading account:

```
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
      Comment("Trading is forbidden for the account ",AccountInfoInteger(ACCOUNT_LOGIN
            ".\n Perhaps an investor password has been used to connect to the trading
            "\n Check the terminal journal for the following entry:",
```

```
            "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': trading has been disabled -
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED) may return **false** in the following cases:

- no connection to the trade server. That can be checked using TerminalInfoInteger(TERMINAL_CONNECTED);
- trading account switched to read-only mode (sent to the archive);
- trading on the account is disabled at the trade server side;
- connection to a trading account has been performed in Investor mode.

See also

Client Terminal Properties, Account Properties, Properties of a Running MQL5 Program

Client Terminal Events

Init

Subito dopo che il terminale client carica un programma (un Expert Advisor o indicatore personalizzato) ed avvia il processo di inizializzazione delle variabili globali, l'evento Init verrà inviato, e verrà elaborato dall' event handler OnInit(), se esiste. Questo evento viene anche generato dopo che un strumento finanziario e/o timeframe, del grafico, cambiano, dopo che un programma viene ricompilato in MetaEditor, dopo che i parametri di input vengono modificati dalla finestra di impostazione di un Expert Advisor o di un indicatore personalizzato. Un Expert Advisor viene anche inizializzato dopo che l'account è stato modificato. L'evento Init non viene generato per gli script.

Deinit

Prima che le variabili globali vengano deinizializzate, ed il programma (Expert Advisor o indicatore personalizzato) viene scaricato, il terminale client invia l'evento Deinit al programma. Deinit viene generato anche quando il terminale client viene chiuso, quando un grafico viene chiuso, proprio prima che lo strumento finanziario e/o il timeframe vengono cambiati, o ad una ricompilazione con successo del programma, o quando i parametri di input vengono cambiati, o quando cambia un account.

Le motivazioni di deinizializzazione possono essere ottenute dal parametro, passato alla funzione OnDeinit(). La l'esecuzione della funzione OnDeinit() è limitata a 2,5 secondi. Se durante questo tempo la funzione non è stata completata, viene interrotta prematuramente. L'evento Deinit non viene generato per gli script.

Start

Start è un evento speciale per il lancio di uno script o di un servizio dopo averlo caricato. È gestito dalla funzione OnStart. L'evento Start non viene passato agli EA e agli indicatori personalizzati.

NewTick

L'evento NewTick viene generato se ci sono nuove quotazioni, e viene elaborato da OnTick() dell' Expert Advisor allegato. Nel caso in cui la funzione OnTick per la quotazione precedente è in fase di elaborazione quando viene ricevuta una quotazione, la nuova quotazione verrà ignorata dall' Expert Advisor, perché l'evento corrispondente non sarà messo in fila.

Tutte le nuove citazioni che vengono ricevute mentre il programma è in esecuzione vengono ignorate fino a quando onTick() non è completato. Dopo di che la funzione girerà solo dopo che viene ricevuta una nuova quotazione. L'evento NewTick viene generato indipendentemente dal fatto che sia consentito il trading automatizzato o meno (bottone "permetti/vieta il trading automatizzato"). Il divieto di trading automatico indica solo che l'invio delle richieste di trading provenienti da un Expert Advisor non sono consentite, mentre l'Expert Advisor continua a lavorare.

Il divieto di trading automatico, premendo il relativo bottone, non fermerà l'esecuzione corrente della funzione onTick().

Calculate

L'evento Calculate viene generato solo per gli indicatori giusto dopo che l'evento Init viene inviato ad ogni cambio di dati sui prezzi. Viene elaborato dalla funzione OnCalculate.

Timer

L'evento Timer viene periodicamente generato dal terminale client per l'Expert Advisor che ha attivato il timer della funzione EventSetTimer. Di solito, questa funzione viene chiamata da OnInit. L'evento di elaborazione Timer viene eseguito dalla funzione OnTimer. Dopo che l'operazione dell' Expert Advisor è stata completata, è necessario distruggere il timer con la funzione EventKillTimer, che di solito è chiamata nella funzione OnDeinit.

Trade

L'evento di trade viene generato quando un operazione di trade è stata completata sul trade server. L'evento Trade è gestito dalla funzione OnTrade() per le operazioni di trade seguenti:

- l'invio, la modifica o la rimozione di un ordine pendente;
- l' annullamento di un ordine pendente con non abbastanza denaro o espirazione;
- attivazione di un ordine pendente;
- apertura, aggiunta o chiusura di una posizione (o parte della posizione);
- la modifica della posizione aperta (cambio degli stops - Stop Loss e/o Take Profit).

TradeTransaction

Quando si eseguono alcune azioni precise su un trade account, il suo stato cambia. Tali azioni comprendono:

- Inviare una richiesta di trade da qualsiasi applicazione MQL5 nel terminale client utilizzando le funzioni OrderSend e OrderSendAsync e la sua ulteriore esecuzione;
- Inviare una richiesta di trade tramite l'interfaccia grafica del terminale e la sua esecuzione ulteriore;
- Attivazione di ordini pendenti ed ordini di stop sul server;
- Esecuzione di operazioni sul lato trade server.

Le operazioni commerciali di seguito vengono eseguite come risultato di queste azioni:

- gestione di una richiesta di trade;
- cambio di ordini aperti;
- cambio della cronistoria degli ordini;
- cambio della cronistoria degli affari;
- cambio delle posizioni.

Per esempio, quando si invia un ordine di mercato buy, esso viene gestito, un appropriato ordine di buy viene creato per l'account, l'ordine viene poi eseguito e rimosso dalla lista di quelli aperti, quindi viene aggiunto alla cronistoria ordini, un appropriato affare si aggiunge alla cronistoria e una nuova posizione viene creata. Tutte queste azioni sono transazioni di trade. L' arrivo di una tale operazione al terminal è un evento TradeTransaction. Questo evento viene gestito dalla funzione OnTradeTransaction.

Tester

L'evento Tester viene generato dopo che il testing di un Expert Advisor su dati storici è finito. L'evento è gestito dalla funzione OnTester().

TesterInit

L'evento TesterInit viene generato con l'inizio dell' ottimizzazione nello strategy tester prima del primo step di ottimizzazione. L'evento TesterInit viene gestito dalla funzione OnTesterInit().

TesterPass

L'evento TesterPass viene generato quando viene ricevuto un nuovo frame di dati. L'evento TesterPass viene gestito dalla funzione OnTesterPass().

TesterDeinit

L' evento TesterDeinit viene generato dopo la fine dell' ottimizzazione di un Expert Advisor nello strategy tester. L'evento TesterDeinit viene gestito dalla funzione OnTesterDeinit().

ChartEvent

L' eventoChartEvent viene generato dal terminale client quando un utente sta lavorando con un grafico:

- pressione della tastiera, quando la finestra del grafico è selezionata;
- oggetto grafico creato
- oggetto grafico eliminato
- pressione del tasto del mouse sull'oggetto grafico del grafico
- spostamento dell'oggetto grafico utilizzando il mouse
- fine della modifica del testo nella LabelEdit.

Inoltre vi è un ChartEvent evento personalizzato, che può essere inviato ad un Expert Advisor qualunque utilizzando la funzione EventChartCustom. L'evento viene elaborato dalla funzione OnChartEvent.

BookEvent

L'evento BookEvent viene generato dal terminale client dopo che la Profondità di Mercato è cambiata; viene elaborato dalla funzione OnBookEvent. Per avviare la generazione di BookEvent per il simbolo specificato, è necessario registrare il simbolo per questo evento utilizzando la funzione MarketBookAdd.

Per annullare l'iscrizione a BookEvent per un simbolo specificato, è necessario chiamare la funzione MarketBookRelease. L'evento BookEvent è un evento di tipo-broadcasting - significa che è sufficiente iscrivere un solo Expert Advisor per questo evento, e tutti gli altri Expert Advisor che hanno l'handler event OnBookEvent, lo riceveranno. Ecco perché è necessario analizzare il nome del simbolo, che è passato ad un handler come parametro.

Vedi anche

Event handlers, Funzionamento del programma

Risorse

Uso di immagini e suoni in programmi mql5

I programmi in MQL5 permettono di lavorare con i file audio e grafica:

- PlaySound() riproduce un file audio;
- ObjectCreate() permette di creare interfacce utente utilizzando gli oggetti grafici OBJ_BITMAP e OBJ_BITMAP_LABEL.

PlaySound()

Esempio di chiamata della funzione PlaySound():

```
//+------------------------------------------------------------------------------+
//| Chiama OrderSend() standard e riproduce un suono                             |
//+------------------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
//--- Invia una richiesta ad un server
   OrderSend(request,result);
//--- se la richiesta viene accettata, riproduce l'audio Ok.wav
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
//--- se fallisce, riproduce l' allarme dal file timeout.wav
   else PlaySound("timeout.wav");
  }
```

L'esempio mostra come riprodurre i suoni di 'Ok.wav' file e 'timeout.wav', che sono inclusi nel pacchetto terminale standard. Questi file si trovano nella cartella **terminal_directory\Sounds**. Qui **terminal_directory** vi è la cartella, da cui viene avviato Terminale Client MetaTrader 5. La posizione della directory del terminale può essere scoperta da un programma MQL5 nel modo seguente:

```
//--- Cartella, in cui sono memorizzati i dati del terminale
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

È possibile utilizzare i file audio non solo dalla cartella **terminal_directory\Sounds**, ma anche da qualsiasi sottocartella che si trova **terminal_data_directory\MQL5.** Si può scoprire il percorso della directory dei dati del terminale dal menu del terminale "File" -> "Apri" dati del terminale o con il metodo del programma:

```
//--- Cartella, in cui sono memorizzati i dati del terminale
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Ad esempio, se il file audio Demo.wav si trova nella terminal_data_directory\MQL5\Files, allora la chiamata di PlaySound() dovrebbe essere scritta nel modo seguente:

```
//--- suona Demo.wav dalla cartella terminal_directory_data\MQL5\Files\Demo.wav
   PlaySound("\\Files\\Demo.wav");
```

Si prega di notare che nel commento il percorso del file è scritto usando una barra inversa "\", e nella funzione invece viene usato "\\" .

Quando si specifica il percorso, usare sempre e solo la doppia barra rovesciata come separatore, in quanto una singola barra rovesciata è un simbolo di controllo per il compilatore quando si tratta di stringhe costanti e caratteri costanti nel codice sorgente del programma.

Chiama la funzione PlaySound() con parametro NULL per fermare la riproduzione:

```
//--- la chiamata di PlaySound() con il parametro NULL ferma la riproduzionne
   PlaySound(NULL);
```

ObjectCreate()

Esempio di un Expert Advisor, che crea un'etichetta grafica (OBJ_BITMAP_LABEL) utilizzando la funzione ObjectCreate().

```
string label_name="currency_label";         // nome dell'oggetto OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";     // percorso del file terminal_data_directo
string dollar    ="\\Images\\dollar.bmp";   // percorso del file terminal_data_directo
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- crea un bottone OBJ_BITMAP_LABEL, se non è stato ancora creato
   if(ObjectFind(0,label_name)<0)
     {
//--- cerca di creare l'oggetto OBJ_BITMAP_LABEL
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- link sul pulsante nell'angolo in alto a sinistra del grafico
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
         //--- ora imposta le proprietà di un oggetto
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- reimpostare il codice dell'ultimo errore a 0
         ResetLastError();
//--- scaricare un'immagine per indicare lo stato "Premuto" del bottone
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- testa il risultato
         if(!set)
           {
PrintFormat("Impossibile scaricare l'immagine dal file %s. Error code %d",euro,GetLast
           }
         ResetLastError();
         //--- scarica un'immagine per indicare lo stato "Non premuto" del bottone
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
```

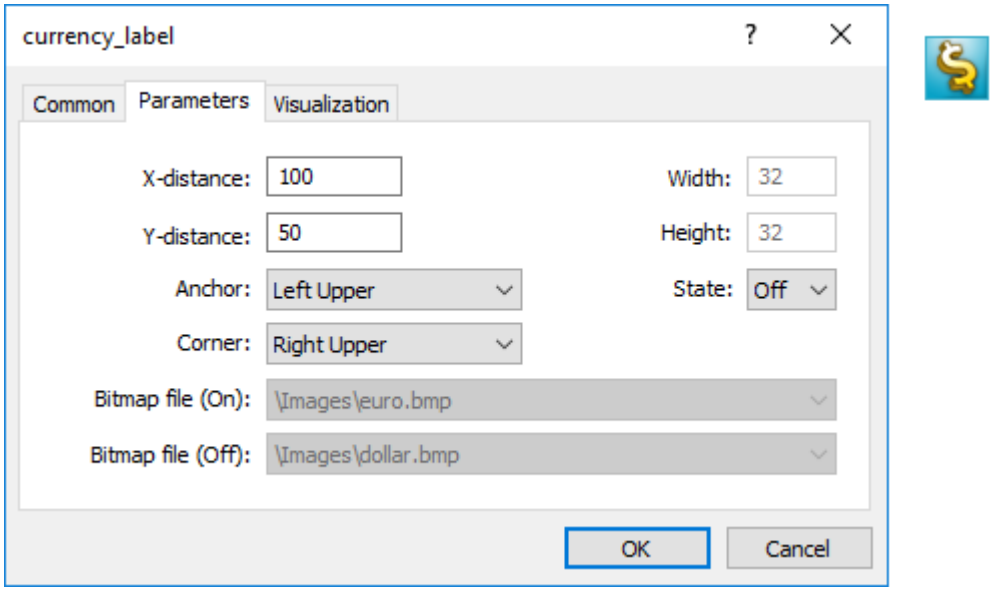
```
            {
PrintFormat("Impossibile scaricare l'immagine dal file %s. Error code %d",dollar,GetLa
            }
//--- invia un comando per un grafico per aggiornare in modo che il bottone viene visu
         ChartRedraw(0);
        }
      else
        {
         // --- fallimento nel creare un oggetto, notifica
         PrintFormat("Fallimento nel creare OBJ_BITMAP_LABEL. Error code %d",GetLastEr
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                                                 |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- elimina un oggetto da un grafico
   ObjectDelete(0,label_name);
  }
```

La creazione e configurazione dell'oggetto grafico denominato currency_label è svolta nella funzione OnInit(). I percorsi dei file grafici si trovano nelle variabili globali**euro** e **dollaro**, una doppia backslash viene utilizzata come separatore:

```
string euro   ="\\Images\\euro.bmp";       // percorso al file terminal_data_directory\
string dollar ="\\Images\\dollar.bmp";     // percorso al file terminal_data_directory\
```

I file si trovano nella cartella **terminal_data_directory**\MQL5\Images.

L'oggetto OBJ_BITMAP_LABEL è in realtà un bottone, che visualizza una delle due immagini, a seconda dello stato del bottone (premuto o non premuto): euro.bmp o dollar.bmp.



La grandelzza del bottone con un'interfaccia grafica viene automaticamente adattata alla dimensione dell'immagine. L'immagine viene modificata con un clic del tasto sinistro del mouse sull'oggetto OBJ_BITMAP_LABEL (**deve essere impostata nelle proprietà "Disattivare la selezione"**). L'oggetto OBJ_BITMAP viene creato nello stesso modo - viene utilizzato per creare lo sfondo con un'immagine necessaria.

Il valore della proprietà OBJPROP_BMPFILE, che è responsabile per l'aspetto degli oggetti OBJ_BITMAP e OBJ_BITMAP_LABEL, può essere modificato in modo dinamico. Questo permette di creare diverse interfacce utente interattive per programmi mql5.

Comprese le risorse per i file eseguibili durante la compilazione dei programmi mql5

Un programma mql5 può necessitare di un sacco di differenti risorse scaricabili sotto forma di file immagine ed audio. Al fine di eliminare la necessità di trasferire tutti i file quando si sposta un file eseguibile in MQL5, la direttiva del compilatore #resource deve essere utilizzata:

```
#resource percorso_al_file_risorsa
```

Il comando #resource indica al compilatore che la risorsa nel percorso specificato **path_to_resource_file** dovrebbe essere inclusa nel file eseguibile EX5. Così tutte le immagini e suoni necessari possono essere posto direttamente in un file EX5, in modo che non vi sia alcuna necessità di trasferire separatamente i file utilizzati in esso, se si desidera eseguire il programma su un terminale diverso. Qualsiasi file EX5 può contenere le risorse, e qualsiasi programma EX5 può utilizzare le risorse da un altro programma EX5.

I file in formato BMP e WAV vengono automaticamente compressi prima di includerli in un file EX5. Questo indica che in aggiunta alla creazione di programmi completi in MQL5, l'utilizzo di risorse consente inoltre di ridurre la grandezza totale dei file necessari utilizzando immagini e suoni, rispetto al solito modo di scrittura del programma MQL5.

La grandezza del file di risorse non deve superare i 16 Mb.

Ricerca di risorse specificate da un compilatore

Una risorsa viene inserita con il comando #resource "<percorso ad un file risorsa>"

```
#resource "<percorso_ad_un_file_risorse>"
```

La lunghezza della stringa costante <path_to_resource_file> non deve superare i 63 caratteri.

Il compilatore cerca per una risorsa nel percorso specificato nel seguente ordine:

- se il separatore barra rovesciata "\" (scritto come "\\") è posto all'inizio del percorso, esso ricerca per il resource relativo alla directory **terminal_data_directory**\MQL5\,
- se non c'è backslash, esso cerca la risorsa relativa alla posizione del file di origine, in cui è scritta la risorsa.

Il percorso delle risorse non può contenere le sottostringhe "..\\" e ":\\".

Esempi di integrazione delle risorse:

```
//--- specifiche corrette delle risorse
#resource "\\Images\\euro.bmp" // euro.bmp è localizzato in terminal_data_directory\MQ
#resource "picture.bmp"        // picture.bmp è localizzato nella stessa directory del
#resource "Resource\\map.bmp"  // la risorsa è localizzata nel file sorgente source_fi

//--- specificazione non corretta della risorsa
#resource ":picture_2.bmp"     // non deve contenere ":"
#resource "..\\picture_3.bmp"  // non deve contenere ".."
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" //più d
```

Uso delle risorse

Nome risorsa

Dopo che una risorsa viene dichiarata mediante la direttiva #resource, essa può essere utilizzata in qualsiasi parte di un programma. Il nome della risorsa è il suo percorso senza backslash all'inizio della riga, che imposta il percorso della risorsa. Per usare le tue risorse nel codice, il contrassegno speciale "::" deve essere aggiunto prima il nome della risorsa.

Esempi:

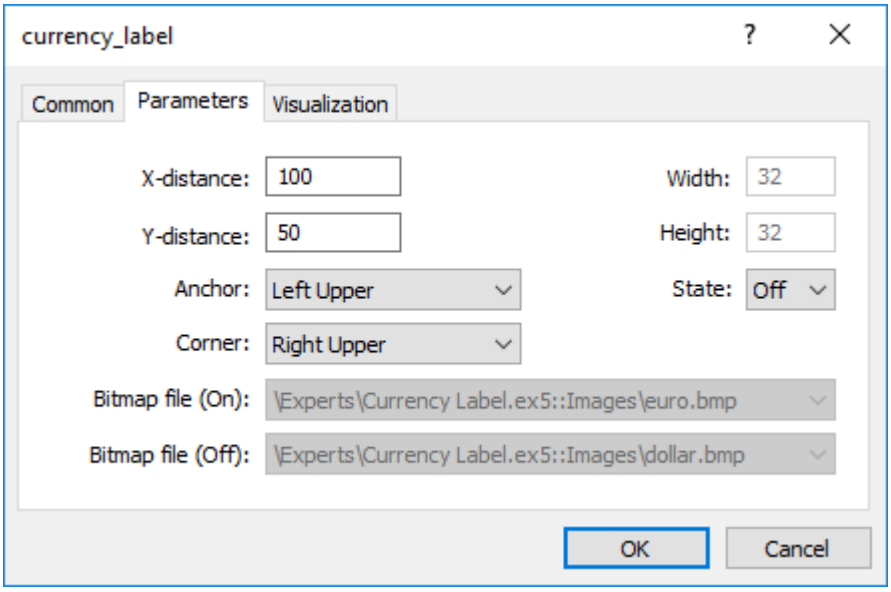
```
//--- esempi di specifiche risorse e dei loro nomi nei commenti
#resource "\\Images\\euro.bmp"          // nome risorsa - Images\euro.bmp
#resource "picture.bmp"                 // nome risorsa - picture.bmp
#resource "Resource\\map.bmp"           // nome risorsa - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // nome risorsa - Files\Pictures\good.bmp
#resource "\\Files\\Demo.wav";          // nome risorsa - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";       // nome risorsa - Sounds\thrill.wav"
...
```

```
//--- utilizzazione delle risorse
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

Occorre notare che quando si impostano immagini da una risorsa agli oggetti OBJ_BITMAP e OBJ_BITMAP_LABEL, il valore della proprietà OBJPROP_BMPFILE non può essere modificato manualmente. Ad esempio, per la creazione di OBJ_BITMAP_LABEL utilizziamo le risorse euro.bmp e dollar.bmp.

```
#resource "\\Images\\euro.bmp";    // euro.bmp si trova in terminal_data_directory\MQ
#resource "\\Images\\dollar.bmp";  // dollar.bmp si trova in terminal_data_directory\
```

Quando si visualizzano le proprietà di questo oggetto, vedremo che la proprietà BitMap File (On) e BitMap File (Off) sono disattivate e non possono essere cambiate manualmente:



Utilizzando le risorse di altri programmi mql5

Vi è un altro vantaggio della risorsa utilizzando - in un qualsiasi programma MQL5, risorse di un altro file EX5 . Così le risorse da un file EX5 possono essere usate in molti altri programmi mql5.

Per utilizzare il nome di risorsa da un altro file, dovrebbe essere specificato come <path_EX5_file_name>::<resource_name> . Ad esempio, si supponga che lo script Draw_Triangles_Script.mq5 contiene una risorsa di un'immagine nel file triangle.bmp:

```
#resource "\\Files\\triangle.bmp"
```

Allora il suo nome, per l'utilizzo nello script stesso, sarà tipo "File\triangle.bmp", e per usarlo, "::" deve essere aggiunto al nome della risorsa.

```
//--- utilizzo risorsa nello script
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

Per poter utilizzare la stessa risorsa da un altro programma, ad esempio da un Expert Advisor, abbiamo bisogno di aggiungere al nome della risorsa il percorso del relativo file da EX5 **terminal_data_directory\MQL5** ed il nome del file EX5 dello script, il EX5 file - **Draw_Triangles_Script.ex5**. Si supponga che lo script si trovi nella cartella standard **terminal_data_directory\MQL5**\Scripts\, dunque la chiamata deve essere scritta come segue:

```
//--- Utilizzando una risorsa da uno script in un EA
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Se il percorso del file eseguibile non è specificato quando si chiama la risorsa da un altro EX5, il file eseguibile viene cercato nella stessa cartella che contiene il programma che chiama la risorsa. Ciò significa che se un Expert Advisor richiede una risorsa da Draw_Triangles_Script.ex5 senza specificazione del percorso, in questo modo:

```
//--- chiamata script di risorsa in un EA senza specificare il percorso
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

allora il file verrà ricercato per la cartella **terminal_data_directory\MQL5**\Experts\, se gli Expert Advisor sono memorizzati in **terminal_data_directory\MQL5**\Experts\.

Funzionamento con indicatore personalizzato incluso come risorsa

Uno o più indicatori personalizzati possono essere necessari per il funzionamento delle applicazioni MQL5. Tutti possono essere inclusi nel codice di un programma MQL5 eseguibile. L'inclusione di indicatori come risorse semplifica la distribuzione delle applicazioni.

Di seguito è riportato un esempio di inclusione ed utilizzazione dell'indicatore personalizzato SampleIndicator.ex5 che si trova in **terminal_data_folder**\MQL5\Indicators\ directory:

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                        |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
   if(handle_ind==INVALID_HANDLE)
```

```
      {
       Print("Expert: iCustom call: Error code=",GetLastError());
       return(INIT_FAILED);
      }
//--- ...
   return(INIT_SUCCEEDED);
  }
```

Il caso quando un indicatore personalizzato nella funzione OnInit() crea una o più copie di se stesso e richiede una speciale considerazione. Si prega di tenere presente che la risorsa deve essere specificata nel seguente modo: **<path_EX5_file_name>::<resource_name>**.

Ad esempio, se l'indicatore SampleIndicator.ex5 è incluso per SampleEA.ex5 Expert Advisor come una risorsa, il percorso a sé specificato al momento della chiamata della iCustom() enlla funzione di inizializzazione dell' indicatore personalizzato appare nel seguente modo: "\\Experts\ \SampleEA.ex5::Indicators\\SampleIndicator.ex5".. Quando questo percorso è impostato in modo esplicito, l'indicatore personalizzato SampleIndicator.ex5 è rigidamente collegato all' Expert Advisor SampleEA.ex5 e perde la capacità di lavorare in modo indipendente.

Il percorso di per sé può essere ricevuto tramite la funzione GetRelativeProgramPath(). L'esempio del suo utilizzo è il seguente:

```
//+------------------------------------------------------------------+
//|                                                SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- il modo sbagliato di fornire un collegamento a se stesso
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- il modo giusto per ricevere un collegamento a se stesso
  string path=GetRelativeProgramPath();
//--- mappatura buffers indicatore
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("Indicator: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
   else Print("Indicator handle=",handle);
//---
   return(INIT_SUCCEEDED);
  }
```

```
///....
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
   int pos2;
//--- ottiene il percorso assoluto per l'applicazione
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- trova la posizione della sottostriga "\MQL5\"
   int    pos =StringFind(path,"\\MQL5\\");
//--- sottostringa non trovata - errore
   if(pos<0)
      return(NULL);
//--- salta la directory "\MQL5"
   pos+=5;
//--- salta il simbolo '\'
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- se c'è una risorsa, restituisce il percorso relativo alla directory MQL5
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- trova un separatore per la prima sottodirectory MQL5 (per esempio, MQL5\Indicato
//--- se non trovato, restituisce il percorso relativo alla directory MQL5
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- restituisce il percorso relativo alla sottodirectory (per esempio, MQL5\Indicato
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Variabili di risorsa

Le risorse possono essere dichiarate utilizzando le variabili di risorsa e trattati come se fossero variabili del tipo appropriato. Formato Dichiarazione:

```
#resource percorso_al_file_risorsa as tipo_variabile_risorsa nome_variabile_risorsa
```

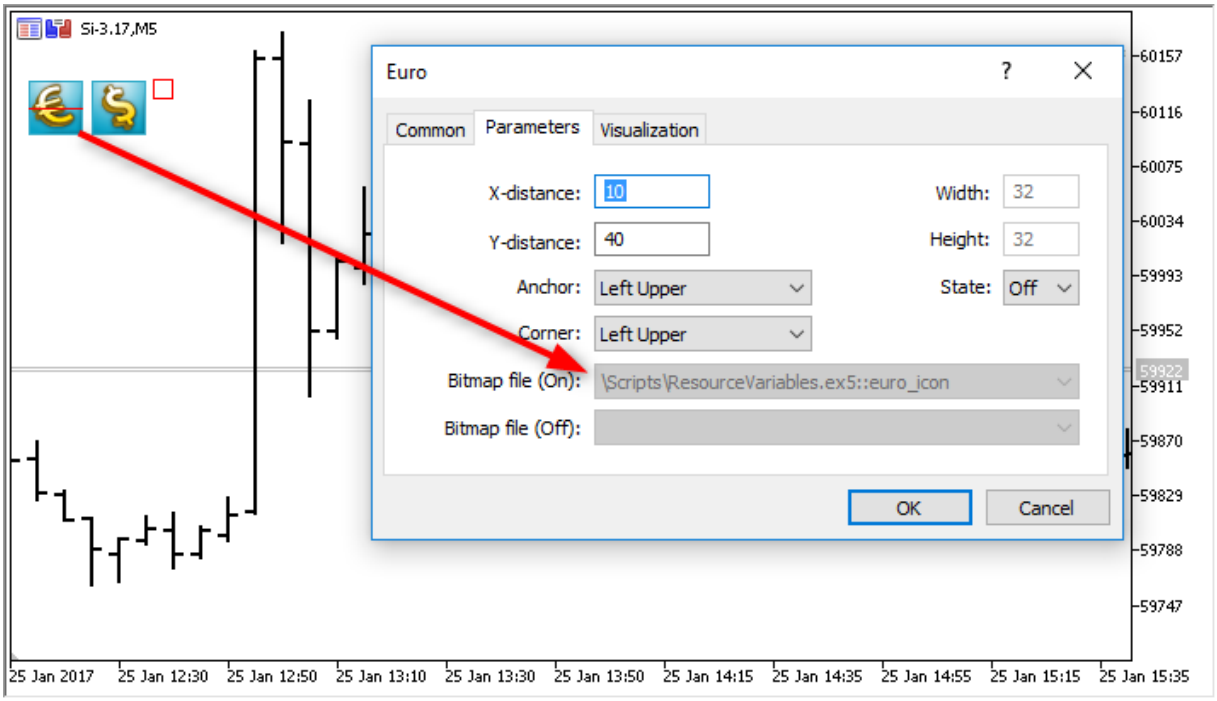
Dichiarazioni di esempio:

```
#resource "data.bin" as int ExtData[]              // dichiara l'array numerico contene
#resource "data.bin" as MqlRates ExtData[]         // dichiara l'array strutture sempli
//--- strings
#resource "data.txt" as string ExtCode             // dichiara la stringa contenente i
#resource "data.txt" as string ExtCode[]           // dichiara l'array contenente le st
// --- risorse grafiche
#resource "image.bmp" as bitmap ExtBitmap[]        // dichiara l'array mono-dimensional
#resource "image.bmp" as bitmap ExtBitmap2[][]     // dichiara l'array bi-dimensionale
```

In caso di tale dichiarazione, la risorsa dati può essere indirizzata solo attraverso la variabile, l'auto indirizzamento via "::<rsource name>" non funziona.

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//|  OBJ_BITMAP_LABEL funzione creazione oggetto usando la risorsa   |
//+------------------------------------------------------------------+
void Image(string name,string rc,int x,int y)
  {
   ObjectCreate(0,name,OBJ_BITMAP_LABEL,0,0,0);
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
   ObjectSetString(0,name,OBJPROP_BMPFILE,rc);
  }
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grandezza output dell'immagine [larghezza, altezza] memorizzata nella variabile
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- cambia l'immagine in euro - disegna la riga orizzontale red horizontal nel mezzo
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- crea la risorsa grafica utilizzando la variabile risorsa
   ResourceCreate("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRang
//--- Crea l'oggetto grafico etichetta Euro, a cui verrà impostata l'immagine dalla ri
   Image("Euro","::euro_icon",10,40);
//--- un altro metodo di applicazione della risorsa, non si può disegnare su esso
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- il metodo diretto di indirizzare la risorsa euro.bmp non è disponibile in quanto
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // il tempo di esecuzi
  }
```

Risultato di esecuzione Script - solo due OBJ_BITMAP_LABEL oggetti su tre sono stati creati. L'immagine del primo oggetto ha la striscia rossa nel mezzo.



Un importante vantaggio di applicare le risorse è che i file di risorse vengono compressi automaticamente prima di essere inclusi in un file eseguibile EX5 prima della compilazione. Così, l'uso delle variabili di risorse consente di mettere tutti i dati necessari direttamente nel file eseguibile EX5, nonché ridurre il numero e la dimensione totale dei file rispetto al modo tradizionale di scrivere programmi MQL5.

L'utilizzo delle variabili di risorse è particolarmente conveniente per i prodotti editoriali nel Market.

Caratteristiche

- La speciale risorsa variabile di tipo *bitmap*informa il compilatore che la risorsa è un'immagine. Tali variabili ricevono il tipo uint.
- L'array del tipo di variabile risorsa *bitmap*può avere due dimensioni. In questo caso, la dimensione dell'array è definita come [altezza_immagine] [larghezza_immagine]. Se non viene specificato un array ad una dimensione, il numero di elementi è pari ad altezza_immagine * larghezza_immagine.
- Durante il download di immagini a 24 bit, il componente canale alfa è impostato a 255 per tutti i pixel dell'immagine.
- Quando si scarica un'immagine a 32 bit senza il canale alfa, la componente canale alfa è anche impostata a 255 per tutti i pixel dell'immagine.
- Quando si scarica un'immagine a 32 bit con il canale alfa, i pixel non vengono elaborati in alcun modo.
- La grandezza del file di risorse non può superare i 128 Mb.
- Il rilevamento automatico di codifica presenza da BOM (header) viene eseguita per i file di stringa. Se BOM è assente, la codifica è definita dal contenuto del file. Sono supportati i file nei ANSI, UTF-8 e UTF-16. Tutte le stringhe vengono convertite in Unicode durante la lettura dei dati dai file.

OpenCL programs

L'utilizzo delle variabili stringa risorsa può facilitare notevolmente lo sviluppo di alcuni programmi. Ad esempio, si è in grado di scrivere un codice di un programma OpenCL in un file CL separato e poi includerlo come una stringa nella vostre risorse MQL5 del programma.

```
#resource "seascape.cl" as string cl_program
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- esegue ulteriori azioni, con un programma OpenCL
  }
```

In questo esempio, si sarebbe dovuto scrivere l'intero codice come un'unica grande stringa se non sono stati usati variabili risorse*cl_program*.

Vedi anche

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), Funzioni con i Files

Chiamata delle funzioni importate

Per importare le funzioni durante l'esecuzione di un programma MQL5, il terminale client utilizza l'associazione anticipata. Ciò significa che se un programma ha chiamata di una funzione importata, il modulo corrispondente (ex5 o dll) viene caricato durante il caricamento del programma. MQL5 e librerie DLL vengono eseguite nel thread di un modulo chiamante.

Si raccomanda di non utilizzare il nome completo specificato del modulo da caricare come *Unità:\Directory\FileName.Ext*. Le librerie MQL5 vengono caricate dalla cartella *terminal_dir\MQL5\Libraries*. Se la libreria non è stata trovata, il terminale client esegue un tentativo di caricamento dalla cartella *terminal_dir\experts*.

Le librerie di sistema (DLL) vengono caricatee dalle regole del sistema operativo. Se la libreria è già caricata (per esempio, un altro Expert Advisor, e anche da un altro terminale client, in parallelo), allora utilizza richieste alla libreria già caricata. Altrimenti, esegue una ricerca nella seguente sequenza:

1. La directory, da cui il modulo importazione dll è stato avviato. Il modulo qui è un Expert Advisor, uno script, un indicatore o una libreria ex5;
2. Directory terminal_data_directory\MQL5\Libraries (TERMINAL_DATA_PATH\MQL5\Libraries);
3. Directory, da cui è stato avviato il terminale client MetaTrader 5;
4. Directory di sistema
5. Directory di Windows;
6. Directory corrente;
7. Directory elencate nella variabile di sistema PATH.

Se la libreria DLL utilizza un'altra DLL nel suo lavoro, la prima non può essere caricata nel caso in cui non vi è alcuna seconda DLL.

Prima che un Expert Advisor (script, direzione) venga caricato, viene formata una lista comune di tutti i moduli di libreria EX5. Va utilizzato sia da un Expert Advisor caricato (script, indicatore) che da librerie di questa lista. Quindi è necessario caricamento, una volta, di moduli EX5 utilizzati molte volte. Le librerie utilizzano le variabili predefinite degli Expert Advisor (script, indicatore) dal quale sono state chiamate.

La libreria importata EX5 viene cercata nella seguente sequenza:

1. Directory, percorso che è impostato relativelativamente alla directory di Expert Advisor (script, indicatore) che importa EX5);
2. Directory terminal_directory\MQL5\Libraries;
3. Directory MQL5\Libraries nella directory comune a tutti i terminali client di MetaTrader 5 (Common\MQL5\Libraries).

Funzioni importate DLL in un programma-mql5 devono garantire l'accordo delle chiamate Windows API. Per garantire tale accordo, nel testo sorgente dei programmi scritti in C o C++, utilizzare la parola chiave __ stdcall, che è specifico per i compilatori Microsoft(r). Questo accordo è caratterizzato dal seguente:

- il chiamante (nel nostro caso è un programma-mql5) dovrebbe "vedere" un prototipo di una funzione chiamata (importata dalla DLL), al fine di combinare opportunamente i parametri ad uno stack;

- il chiamante (nel nostro caso è un programma mql5) mette i parametri nello stack in ordine inverso, da destra a sinistra - in questo modo una funzione importata legge i parametri passati ad essa;
- i parametri vengono passati per valore, ad eccezione di quelli esplicitamente passati per riferimento (nel nostro caso le stringhe)
- una funzione importata pulisce lo stack in modo indipendente con la lettura dei parametri passati.

Quando si descrive il prototipo di una funzione importata, possono essere utilizzati i parametri di default.

Se la libreria corrispondente non è in grado di caricare, o vi è un divieto di utilizzare la DLL o la funzione di importazione non è stata trovata - l'Expert Advisor arresta il suo funzionamento con l'apposito messaggio "Expert Advisor fermato" nella Journal (file di log). In questo caso l'Expert Advisor non verrà eseguito fino a quando non viene reinizializzato. Un Expert Advisor può essere reinizializzato a seguito di ricompilazione o dopo che la tabella delle sue proprietà si apre e si preme OK.

Passaggio di parametri

Tutti i parametri di tipi semplici vengono passati per valori a meno che non sia espressamente indicato che siano passati per riferimento. Quando viene passata una stringa, viene passato l'indirizzo del buffer della stringa copiata, se una stringa viene passata per riferimento, l'indirizzo del buffer di questa stringa senza copiarla, viene passato alla funzione importata da DLL.

Le Strutture che contengono gli array dinamici, le stringhe, le classi, le altre strutture complesse, così come array dinamici o statici degli oggetti enumerati, non possono essere passati come parametro ad una funzione importata.

Quando si passa un array ad una DLL, l'indirizzo di inizio del buffer di dati viene sempre passato (indipendentemente dal flag AS_SERIES). Una funzione all'interno di una DLL non sa nulla della bandiera AS_SERIES, l'array passato è un array statico di una lunghezza indefinita, dovrebbe essere utilizzato un parametro aggiuntivo per specificare la dimensione dell'array.

Errori di Runtime

Il sottosistema di esecuzione del terminale client ha la possibilità di salvare il codice di errore in caso si verifichi durante il funzionamento di un programma MQL5. Vi è una variabile predefinita _LastError per ogni programma MQL5 eseguibile.

Prima di iniziare la funzione OnInit, la variabile _LastError viene resettata. In caso si verifichi una situazione errata durante calcoli o in fase di chiamate di funzione interne, la variabile _LastError accetta un codice di errore corrispondente. Il valore memorizzato in questa variabile può essere ottenuto usando la funzione GetLastError().

Ci sono diversi errori critici nel caso in cui si verifichino, il programma viene terminato immediatamente:

- divisione per zero
- andare oltre il confine dell'array
- utilizzando uno scorretto puntatore ad oggetto

Testare Strategie di Trading

L'idea di trading automatico è attraente per il fatto che il robot di trading è in grado di lavorare ininterrottamente per 24 ore al giorno, sette giorni alla settimana. Il robot non si stanca, non ha dubbi o paure, è totalmente esente da problemi psicologici. E' sufficiente formalizzare in modo chiaro le regole di trading e la loro attuazione negli algoritmi, ed il robot è pronto a lavorare senza sosta. Ma prima, è necessario assicurarsi che le seguenti due condizioni importanti siano soddisfatte:

- L'Expert Advisor esegue operazioni di trading secondo le regole del sistema di trading;
- La strategia di trading, implementata nell'EA, dimostra un profitto sulla cronistoria.

Per avere risposte a queste domande, ci rivolgiamo allo Strategy Tester, incluso nel terminale client MetaTrader 5.

Questa sezione illustra le caratteristiche del programma di testing ed ottimizzazione nello strategy tester:

- Limiti di funzione nel Tester di Strategia
- Modalità Generazione Tick
- Simulazione dello spread
- Utilizzare ticks reali durante un test
- Le variabili globali del Terminale Client
- Il Calcolo degli Indicatori durante il Testing
- Caricamento dello Storico durante il Testing
- Testing Multi-valuta
- Simulazione del Tempo nello Strategy Tester
- Gli Oggetti Grafici nel Testing
- La funzone OnTimer() nello Strategy Tester
- La funzione Sleep() nello Strategy Tester
- Utilizzo dello Strategy Tester per i Problemi di Ottimizzazione in Calcoli Matematici
- La sincronizzazione delle Barre nella modalità "Solo prezzi di apertura"
- La funzione IndicatorRelease() nella Tester
- Event Handling nel Tester
- Agenti Testing
- Lo Scambio di Dati tra il Terminale e l'Agente
- Uso della Cartella Condivisa di tutti i Terminali Client
- Utilizzo di DLLs

Limiti di memoria e spazio su disco nella MQL5 Cloud Network

La seguente limitazione si applica alle ottimizzazioni eseguite nella MQL5 Cloud Network: l'Expert Advisor non deve scrivere su disco più di 4GB di informazioni o utilizzare più di 4GB di RAM. Se il

limite viene superato, l'agente di rete non sarà in grado di completare correttamente il calcolo e non riceverai il risultato. Tuttavia, ti verrà addebitato tutto il tempo speso per i calcoli.

Se hai bisogno di ottenere informazioni da ogni passaggio di ottimizzazione, invia dei frames senza scrivere sul disco. Per evitare di utilizzare operazioni con i file negli Expert Advisors durante i calcoli nella rete cloud MQL5, è possibile utilizzare il seguente controllo:

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Limiti di Funzione nel Tester di Strategia

Esistono limiti operativi per alcune funzioni nello Strategy Tester del terminale client.

Le funzioni Comment(), Print() e PrintFormat()

Per aumentare le prestazioni, le funzioni Comment(), Print() e PrintFormat() non vengono eseguite durante l'ottimizzazione dei parametri del robot di trading. L'eccezione è l'uso di queste funzioni all'interno dell handler OnInit(). Questo ti permette di trovare facilmente la causa degli errori quando si verificano.

Le funzioni Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest()

Le funzioni Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() e WebRequest() progettate per l'interazione con il "mondo esterno" non vengono eseguite nel tester di strategia.

Modalità Generazione Tick

Un Expert Advisor è un programma, scritto in MQL5, che viene eseguito ogni volta in risposta ad alcuni eventi esterni. La EA ha una corrispondente funzione (event handler) per ogni evento predefinito.

L'evento NewTick (variazione di prezzo) è l'evento principale per l'EA e, di conseguenza, abbiamo bisogno di generare una sequenza di tick per testare l' EA. Ci sono 3 modalità di generazione tick implementate nello Strategy Tester del terminale client di MetaTrader 5:

- Ogni tick
- 1 minuto OHLC (prezzi OHLC con barre minuto)
- Solo prezzi di apertura

La modalità base e la più dettagliata è la modalità "Ogni tick", le altre due modalità sono le semplificazioni della base, e verranno descritte rispetto alla modalità "Ogni tick". Considerate tutte le tre modalità, al fine di capire le differenze tra di loro.

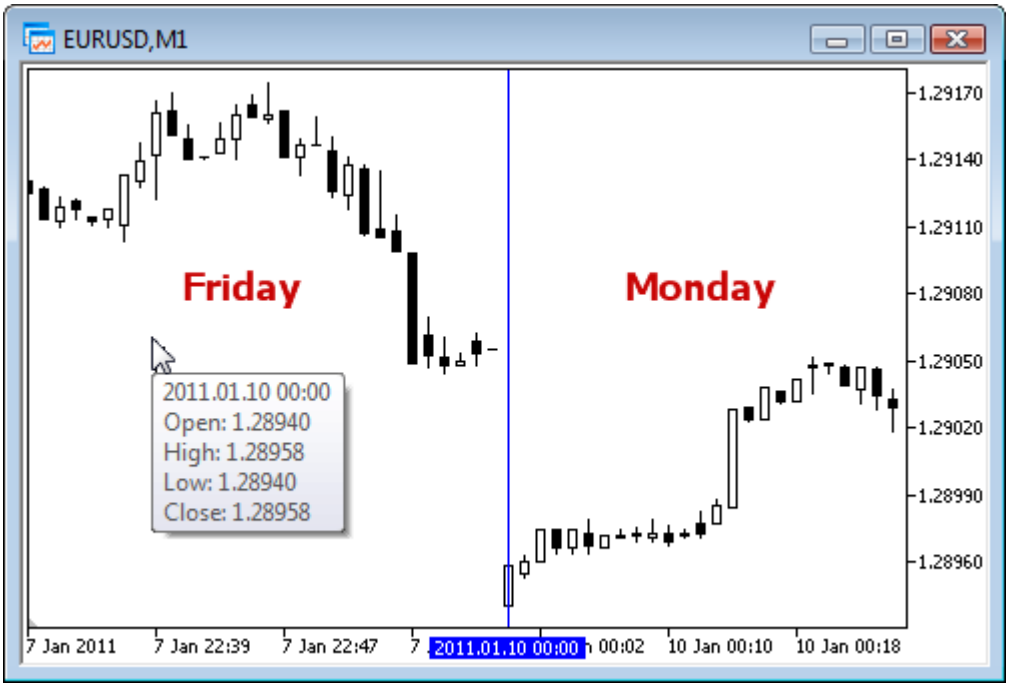
"Every Tick"

I dati storici di quotazioni di strumenti finanziari vengono trasferiti dal trade server al terminale client MetaTrader 5 sotto forma di barre minutamente impacchettate . Informazioni dettagliate sulla presenza delle richieste e la costruzione del timeframe richiesto può essere ottenuta dal capitolo Organizzare Dati di Accesso nel manuale di riferimento MQL5.

L'elemento minimo dello storico dei prezzo è la barra minuta, da cui è possibile ottenere informazioni sui quattro valori del prezzo:

- Open - il prezzo al quale è stata aperta la barra minuta;
- High - il massimo che è stato raggiunto nel corso di questa barra minuta;
- Low - il minimo che è stato raggiunto nel corso di questa barra minuta;
- Close - il prezzo di chiusura della barra.

La nuova barra dei minuti non viene aperta nel momento in cui inizia il nuovo minuto (il numero di secondi diventa uguale a 0), ma quando si verifica un tick - un cambio di prezzo di almeno un punto. La figura mostra la barra primo minuto della nuova settimana di trading, che ha orario di apertura 2011.01.10 00:00. Il divario di prezzo tra Venerdì e Lunedì, che vediamo sul chart, è comune, poiché i tassi di cambio fluttuano anche nei fine settimana in risposta alle notizie in arrivo.

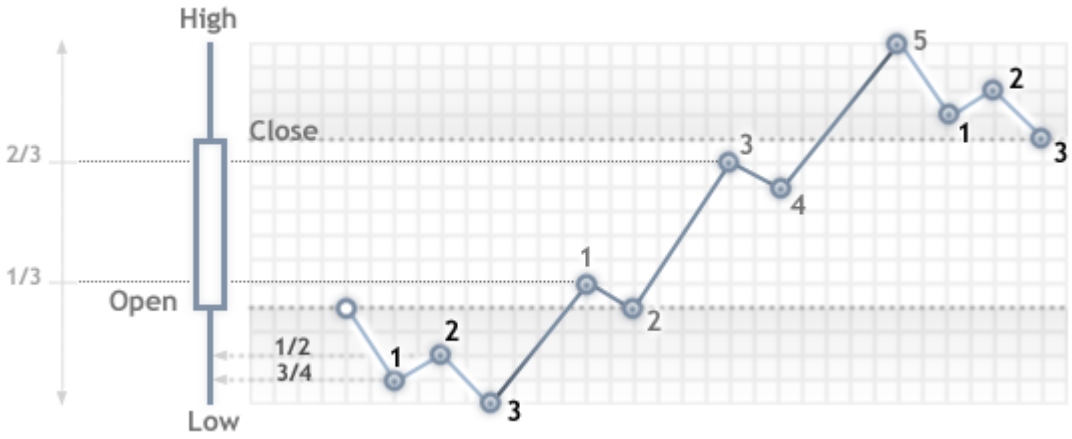


Per questa barra, si sa solo che la barra minuta è stata aperta il 10 gennaio 2011 alle 00 ore 00 minuti, ma non sappiamo nulla circa i secondi. Avrebbe potuto essere aperta alle 00:00:12 o 00:00:36 (12 o 36 secondi dopo l'inizio di un nuovo giorno) o qualsiasi altro momento entro tale minuto. Ma sappiamo che il prezzo di apertura di EURUSD era a 1.28940 al tempo di apertura della barra del nuovo minuto.

Non sappiamo anche, in un secondo, quando il tick corrispondente al prezzo di chiusura della barra minuto considerata, è stato ricevuto. Sappiamo solo una cosa - l'ultimo prezzo Close della barra minuto. Per questo minuto, il prezzo era 1,28958. Il tempo di comparsa di prezzi High e Low è sconosciuto anche, ma sappiamo che i prezzi Minimi(Low) e Massimi(High) sono stati a livello di 1,28958 e 1,28940, rispettivamente.

Per testare la strategia di trading, abbiamo bisogno di una sequenza di ticks, in cui il lavoro dell' Expert Advisor verrà simulato. Così, per ogni barra minuto, sappiamo che i **4 punti di controllo**, dove il prezzo vi è stato sicuramente. Se un bar ha solo 4 ticks, allora ciò è un informazione sufficiente per effettuare un testing, ma di solito il volume di ticke è superiore a 4.

Quindi, vi è la necessità di generare punti di controllo supplementari per i ticks, che si sono verificati tra i prezzi Open, High, Low e Close. Il principio della generazione ticks modalità "Ogni tick" è descritto L'algoritmo di Generazione Ticks all'interno dello Strategia Tester del Terminale MetaTrader 5 del quale una figura è presentata di seguito.



Durante il testing in modalità "Ogni Tick", la funzione OnTick() dell' EA verrà chiamata ad ogni punto di controllo. Ogni punto di controllo è un tick da una sequenza generata. L' EA riceverà l'orario ed il prezzo del tick simulato, così come sarebbe quando si lavora online.

Importante: la modalità testing "Ogni tick" è la più precisa, ma al tempo stesso, quella che richiede più tempo. Per un test iniziale della maggioranza delle strategie di trading, è generalmente sufficiente utilizzare una delle altre due modalità di test.

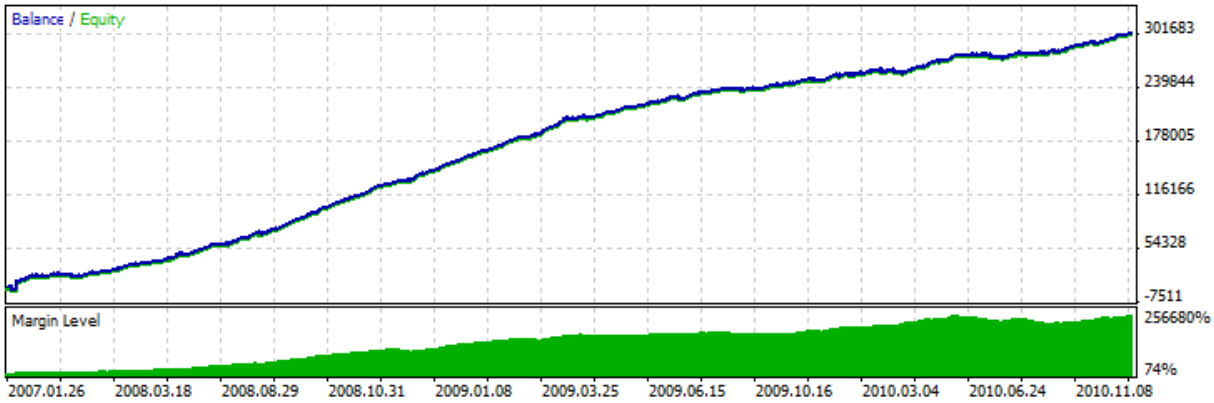
"1 Minuto OHLC"

La modalità "Ogni tick" è la più accurata delle tre modalità, ma al tempo stesso, è la più lenta. La gestione dell'handler OnTick() si verifica ad ogni tick, mentre il volume tick può essere molto grande. Per una strategia, in cui la sequenza ticks dei movimenti dei prezzi in attraverso tutta la barra, non ha importanza, vi è una modalità di simulazione più rapida e più grezza - "1 Minuto OHLC".

Nella modalità "1 minuto OHLC", la sequenza ticks è costruita solo dai **prezzi OHLC delle barre minuto**; il numero dei punti di controllo generati è significativamente ridotto - quindi, e di conseguenza è ridotto il tempo di testing. Il lancio della funzione onTick() viene eseguito su tutti i punti di controllo, che sono costruiti dai prezzi delle barre OHLC minuto.

Il rifiuto di generare ulteriori ticks intermedi tra i prezzi Open, High, Low e Close, porta ad un aspetto di rigido determinismo nello sviluppo dei prezzi, dal momento che viene determinato il prezzo di apertura. Ciò permette di creare un "Testing Graal", che mostra un grafico piacevole di rialzo del saldo testing.

Un esempio di tale Graal è presentato nel Code Base - Grr-al.



La figura mostra un grafico molto interessante di questo EA testing. Come è stato ottenuto? Conosciamo 4 prezzi per una barra minuto, e sappiamo anche che il primo è il prezzo di apertura, e l'ultimo è il prezzo di chiusura. Abbiamo i prezzi Alti e Bassi tra loro, e la sequenza del loro verificarsi è sconosciuta, ma è noto, che il prezzo High(alto) è maggiore o uguale al prezzo di Apertura(Open), ed il prezzo Low(basso) è minore o uguale al prezzo Open.

E' sufficiente per determinare il momento di ricezione del prezzo di apertura(Open), e quindi analizzare il tick successivo al fine di determinare quale prezzo abbiamo in questo momento - sia l'Alto o il Basso. Se il prezzo è inferiore al prezzo Open, allora abbiamo un prezzo Low ed il buy in questo tick; il tick successivo corrisponderà al prezzo High, al quale noi chiudiamo il buy e apririamo per sell. Il tick successivo è l'ultimo, questo è il prezzo Close, e chiudiamo la vendita su di esso.

Se dopo il prezzo, riceviamo un tick con un prezzo superiore al prezzo di apertura, allora la sequenza di affari è invertita. Elabora una barra minuto in questo modo "cheat", ed attende il prossimo.

Durante il testing di questo EA sullo storico, tutto va liscio, ma una volta che lo lanciamo on-line, la verità comincia a rivelarsi - la linea di bilancio rimane costante, ma punta in giù. Per esporre questo trucco, dobbiamo semplicemente eseguire l'EA nella modalità "Ogni tick".

> **Nota:** Se i risultati dell' EA nelle modalità di testing grezze ("OHLC 1 minuto" e "Solo Prezzi di Apertura") sembrano troppo buoni, assicuratevi di provarlo in modalità "Ogni Tick".

"Solo Prezzi di Apertura"

In questa modalità i ticks vengono generati sulla base dei prezzi OHLC del timeframe selezionato per il testing. La funzione OnTick() dell' Expert Advisor gira solo all'inizio della barra al prezzo Open. Grazie a questa caratteristica, i livelli di stop e pendenti, possono innescare un prezzo che differisce da quello specificato (soprattutto nel testing su timeframes superiori). Invece, abbiamo la possibilità di eseguire rapidamente un test di valutazione dell'Expert Advisor.

I periodi W1 e MN1 sono le eccezioni in modalità di generazione ticks "Open Price Only": per questi timeframes i ticks vengono generati per i prezzi OHLC di ogni giorno, non prezzi OHLC della settimana o del mese.

Supponiamo di provare un Expert Advisor su EURUSD H1 nella modalità "Solo Prezzi di Apertura". In questo caso, il numero totale di ticks (punti di controllo) sarà non più di 4*numero di barre un-ora all'interno dell'intervallo testato. Ma l' handler **OnTick() viene chiamato solo all'apertura della barra un-ora**. I controlli richiesti per una analisi corretta si verificano sul resto dei ticks (che sono "nascosti" dalla EA).

- Il calcolo dei requisiti di margine;
- L'innesco di Stop Loss e Take Profit;
- L'innesco di ordini pendenti;
- La rimozione di ordini pendenti espirati.

Se non ci sono posizioni aperte o ordini in sospeso, non abbiamo bisogno di effettuare tali controlli sui ticks nascosti, e l'incremento della velocità può essere sostanzialmente tranquillo. Questa modalità "Solo Prezzi di Apertura" è ben adatta per le strategie di testing, il quale procedimento riguarda solo l'apertura della barra e non utilizza gli ordini pendenti, così come gli ordini StopLoss e TakeProfit. Per la classe di tali strategie, l'accuratezza necessaria del testing viene preservata.

Usiamo l' Expert Advisor Moving Average dal pacchetto astandard, come un esempio di EA, che può essere testato in qualsiasi modalità. La logica di questo EA è costruita in modo tale che tutte le decisioni vengono prese all' apertura della barra, e gli affari vengono eseguiti immediatamente, senza l'uso di ordini pendenti.

Esegue un testing di EA su EURUSD H1 su un intervallo dal 2010.09.01 al 2010.12.31, e confronta i grafici. La figura mostra il grafico saldo dal rapporto di test per tutte le tre modalità.

Come si può vedere, i grafici su diverse modalità di testing sono esattamente gli stessi per l'EA Moving Average dal pacchetto standard.

Ci sono alcune limitazioni sulla modalità "Solo Prezzi di Apertura":

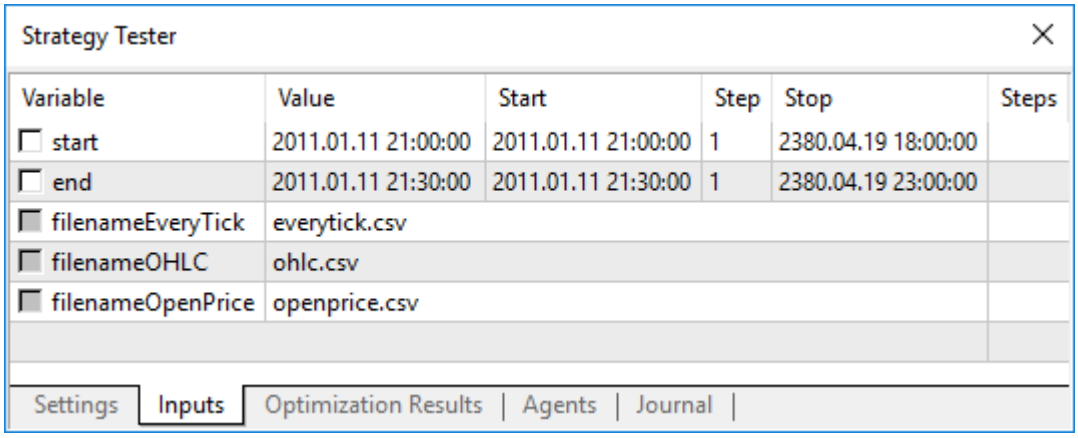
- Non è possibile utilizzare la modalità di esecuzione Random Delay(_* Ritardo Casuale).
- Nella Expert Advisor testato, non è possibile accedere ai dati del timeframe inferiore a quello utilizzato per il testing/ottimizzazione. Ad esempio, se si esegue il testing/ottimizzazione sul periodo H1, è possibile accedere ai dati di H2, H3, H4, ecc., ma non M30, M20, M10, ecc. Inoltre, i timeframes superiori a cui si accede, devono essere multipli del timeframe di testing. Ad esempio, se si esegue il testing in M20, non è possibile accedere ai dati di M30, ma è possibile accedere ad H1. Queste limitazioni sono connesse all'impossibilità di ottenere dati di timeframes inferiori o non multipli su barre generate durante il testing/ottimizzazione.

- Le limitazioni per l'accesso ai dati di altri timeframes si applicano anche ad altri simboli i cui dati vengono utilizzati dall'Expert Advisor. In questo caso la limitazione per ogni simbolo dipende dal timeframe a cui si accede durante il testing/ottimizzazione. Supponiamo che, durante le prove su EURUSD H1, un Expert Advisor accede ai dati di GBPUSD M20. In questo caso l' Expert Advisor sarà in grado di utilizzare ulteriori dati di EURUSD H1, H2, ecc., nonché GBPUSD M20, H1, H2 ecc.

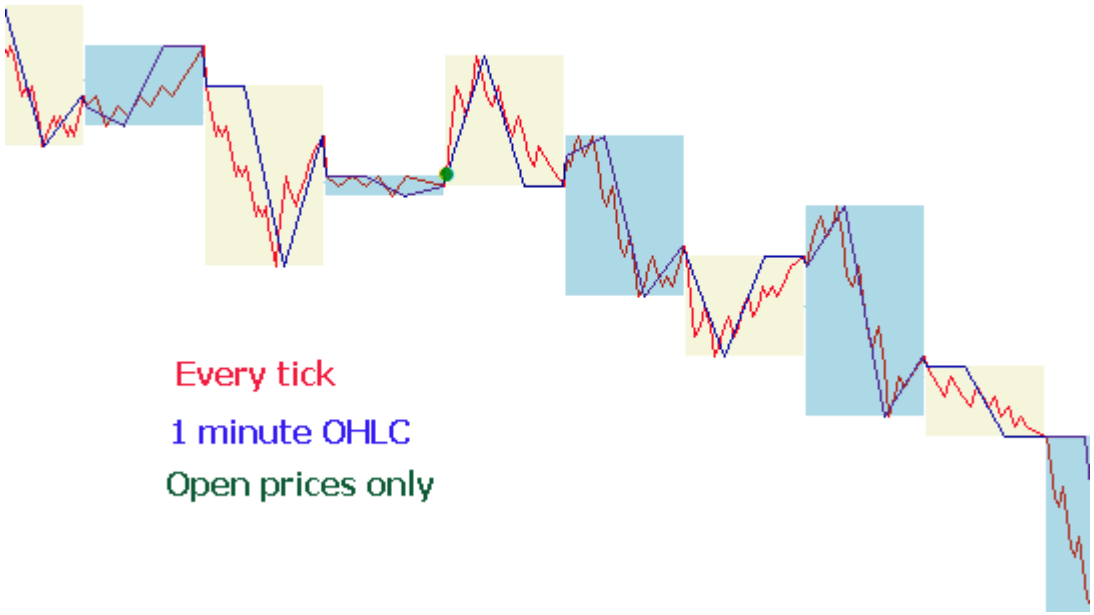
Nota: La modalità "Solo Prezzi di Apertura" ha il più veloce tempo di testing, ma non è adatto per tutte le strategie di trading. Selezionare la modalità di testing desiderata in base alle caratteristiche del sistema di trading.

Per concludere la sezione sulle modalità di generazione dei ticks, consideriamo un confronto visivo delle diverse modalità di generazione di ticks per EURUSD, per due barre M15 su un intervallo da 2011.01.11 21:00:00 - 2011.01.11 21:30:00..

I ticks sono stati salvati in file diversi utilizzando l'EA WriteTicksFromTester.mq5 e la fine di questi nomi file sono specificati in filenameEveryTick, filenameOHLC e filenameOpenPrice (parametri di input).



Per ottenere tre file con tre sequenze di tick (per ciascuna delle seguenti modalità "Ogni Tick", "1 minuto OHLC" e "Solo Prezzi di Apertura"), l'EA è stato lanciato per tre volte nei modi corrispondenti, in runs singole. Quindi, i dati provenienti da questi tre file sono stati visualizzati sul grafico con l'indicatore TicksFromTester.mq5. Il codice dell' indicatore è allegato a questo articolo.



Per default, tutte le operazioni sui file nel linguaggio MQL5 vengono realizzate all'interno del "file sandbox", e durante il testing l'EA ha accesso solo alla propria "sandbox file". Affinché l'indicatore e l' EA possano lavorare con i file da una cartella durante la prova, abbiamo usato il flag FILE_COMMON. Un esempio di codice dall' EA:

```
//--- apre il file
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- controlla l'handle del file
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Errore nell'apertura del file %s per la scrittura. Error code=%d",f
      return;
     }
   else
     {
      PrintFormat("Il file verrà creato nella cartella %s ",TerminalInfoString(TERMINA
     }
```

Per leggere i dati dell'indicatore, abbiamo usato anche il flag FILE_COMMON. Questo ci ha permesso di evitare di trasferire manualmente i file necessari da una cartella ad un'altra.

```
//--- apre il file
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- controlla l'handle del file
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Errore nell'apertura del file %s per la lettura. Error code=%d",fna
      return;
     }
   else
     {
```

```
      PrintFormat("Il file verrà aperto da %s",TerminalInfoString(TERMINAL_COMMONDATA_
     }
```

Simulazione dello spread

La differenza di prezzo tra i prezzi Bid ed Ask si chiama spread. Durante il testing, lo spread non viene modellato, ma è tratto da dati storici. Se lo spread è minore o uguale a zero nei dati storici, allora dall' agente di testing viene utilizzato l'ultimo spread noto (al momento della generazione).

Nello Strategy Tester, lo spread è sempre considerato come fluttuante. Che è SymbolInfoInteger(symobl, SYMBOL_SPREAD_FLOAT) e restituisce sempre true.

Inoltre, i dati dello storico contengono i valori dei tick ed i volumi di trading. Per lo stoccaggio ed il recupero dei dati si usa una speciale struttura MqlRates:

```
struct MqlRates
  {
   datetime time;         // Orario di inizio del periodo
   double   open;         // Prezzo di Apertura(Open)
   double   high;         // Il prezzo Massimo del periodo
   double   low;          // Il prezzo Minimo del periodo
   double   close;        // Prezzo di Chiusura(Close)
   long     tick_volume;  // Volume Tickv
   int      spread;       // Spread
   long     real_volume;  // Volume Trade
  };
```

Utilizzare ticks reali durante un test

Test e ottimizzazione su ticks reali sono il più vicino possibile alle condizioni reali. Invece di ticks generati sulla base dei dati minute, è possibile utilizzare i ticks reali accumulati da un broker. Questi sono i ticks da scambi e fornitori di liquidità.

Per garantire la massima precisione di test, barre minuti vengono utilizzate anche nella modalità ticks reali. Le barre vengono applicate per controllare e correggere i dati tick. Questo permette anche di evitare la divergenza dei charts nel tester e nel terminale client.

Il tester confronta i dati tick con i parametri barra minuto: un tick non dovrebbe superare i livelli Alto/Basso della barra, anche, i ticks iniziali e finali, dovrebbero coincidere con i prezzi Apertura/Chiusura della barra. Il volume viene confrontato pure. Se viene rilevata una mancata corrispondenza, tutti i ticks corrispondenti a tale barra minuto vengono scartati. Vengono invece utilizzati ticks generati(come nella modalità "Ogni tick").

Se uno storico del simbolo ha una barra minuto senza i dati tick per essa, il tester genera ticks in modalità "Ogni tick". Questo permette tracciare un chart corretto nel tester in caso i dati tick di un broker siano insufficienti. Se uno storico del simbolo non dispone di una barra minuto, ma i dati tick appropriati per il minuto sono presenti, i dati possono essere utilizzati nel tester. Ad esempio, le coppie di simboli di scambio vengono formate usando gli Ultimi prezzi. Se solo i ticks con prezzi Bid/Ask, senza l'Ultimo prezzo (Last), arrivano dal server, la barra non viene generata. Il tester utilizza questi dati tick dal momento che non contraddicono quelli al minuto.

I dati tick potrebbero non coincidere con barre minuto per varie ragioni, per esempio a causa di perdite di connessione o altri errori durante la trasmissione dei dati da una sorgente al terminale client. I dati minuto vengono considerati più affidabile durante i test.

Tenete a mente le seguenti caratteristiche durante il test sui ticks reali:

- Quando si lancia un test, i dati aggiornati sul simbolo vengono sincronizzati con quelli tick.
- I tick vengono memorizzati nella cache simbolo dello strategy tester. La dimensione della cache non supera i 128000 ticks. Quando i nuovi ticks arrivano, i dati più vecchi vengono rimossi dalla cache. Tuttavia, la funzione CopyTicks permette la ricezione di ticks fuori della cache (solo quando si fa il test su tick reali). In tal caso, i dati vengono richiesti dal database tick tester che è completamente simile al database terminale client corrispondente. Non vengono implementate correzioni barra minulo a questa base. Pertanto, i ticks possono essere diversi da quelli memorizzati nella cache.

Le variabili globali del Terminale Client

Durante il testing, le variabili globali del terminale client vengono emulate anche, ma non sono collegate alle correnti variabili globali del terminale, che possono essere viste nel terminale utilizzando il tasto F3. Ciò significa che tutte le operazioni con le variabili globali del terminale, durante i testing, hanno luogo al di fuori del terminale client (nell'agente di test).

Il Calcolo degli Indicatori durante il Testing

Nella modalità in tempo reale, i valori dell'indicatore vengono calcolati ad ogni tick.

Nel Tester di strategia, gli indicatori vengono calcolati solo quando si accede ai dati, cioè quando vengono richiesti i valori del buffer dell'indicatore. Le uniche eccezioni sono gli indicatori personalizzati con il #property tester_everytick_calculate specificato. In questo caso, il ricalcolo viene eseguito su ogni tick.

Nella modalità di test visivo, tutti gli indicatori vengono ricalcolati incondizionatamente all'arrivo di un nuovo tick per essere visualizzati correttamente sul grafico di test visivo.

L'indicatore viene calcolato una volta per ogni tick. Tutte le successive richieste di dati dell'indicatore non portano al ricalcolo fino all'arrivo di un nuovo tick. Pertanto, se il timer è abilitato in un EA tramite la funzione EventSetTimer(), i dati dell'indicatore vengono richiesti dall'ultimo tick prima di ogni chiamata del gestore OnTimer(). Se l'indicatore non è stato ancora calcolato sull'ultimo tick, i calcoli dei valori dell'indicatorevengono avviati. Se i dati sono già stati preparati, vengono forniti senza un nuovo ricalcolo.

Pertanto, tutti i calcoli dell'indicatore vengono eseguiti nel modo più efficiente in termini di risorse — se l'indicatore è già stato calcolato a un dato tick, i suoi dati vengono forniti "così come sono". Non viene avviato alcun ricalcolo.

Caricamento dello Storico durante il Testing

Lo storico di un simbolo da testare è sincronizzato e caricato dal terminale, dal trade server prima di avviare il processo di testing. Durante la prima volta, il terminale carica tutto lo storico a disposizione di un simbolo per non richiederlo in un secondo momento. Inoltre solo i nuovi dati vengono caricati.

Un agente di testing riceve lo storico di un simbolo da testare dal terminale client subito dopo l'inizio del testing. Se i dati di altri strumenti vengono utilizzati nel processo di testing (ad esempio, è un Expert Advisor multivaluta), gli agenti di testing richiedono lo storico dal terminale client durante la prima chiamata a tali dati. Se i dati storici sono disponibili nel terminale, vengono immediatamente passati all'agente di testing. Se i dati non sono disponibili, il terminale li richiede e scarica dal server, e poi li passa all'agente di testing.

I dati di strumenti finanziari aggiuntivi sono anche necessari per il calcolo dei cross-rates per le operazioni di trade. Per esempio, quando si fa il testing di una strategia EURCHF con la valuta di deposito in USD, prima di elaborare la prima operazione di trading, l'agente di testing richiede i dati storici di EURUSD e USDCHF dal terminale client, anche se la strategia non contiene la chiamata diretta di uso di questi simboli.

Prima di testare una strategia multi-valuta, si consiglia di scaricare tutti i dati storici necessari nel terminale client. Ciò contribuirà ad evitare ritardi nel testing/ottimizzazione associato con il download dei dati richiesti. È possibile scaricare lo storico, per esempio, aprendo i grafici appropriati e facendo scorrimento verso l'inizio dello storico. Un esempio di caricamento forzato dello storico nel terminale è disponibile nella sezione Organizzare l'Accesso ai Dati dell' MQL5 Reference.

Agenti di testing, a loro volta, ricevono lo storico dal terminale in una forma impacchettata. Durante il testing successivo, il tester non carica lo storico dal terminale, in quanto i dati richiesti sono disponibili dalla precedente esecuzione del tester.

- Il terminale carica lo storico da un trade server sola volta, la prima volta l'agente richiede lo storico di un simbolo da essere testato dal terminale. Lo storico viene caricato in forma impacchettata per ridurre il traffico.
- I ticks non vengono inviati attraverso la rete, sono generati sugli agenti di testing.

Testing Multi-valuta

Lo strategy tester ci permette di eseguire un test di strategie, facendo trading su più simboli. Tali EA sono convenzionalmente indicati come Expert Advisor multi-valuta, dal momento che in origine, nelle piattaforme precedenti, il testing veniva effettuato solo per un unico simbolo. Nello Strategy Tester del terminale MetaTrader 5, possiamo modellare il trading per tutti i simboli disponibili.

Il tester carica lo storico dei simboli utilizzati dal **terminale client** (non dal trade server!) automaticamente durante la prima chiamata dei dati del simbolo.

L'agente di testing scarica solo lo storico mancante, con un piccolo margine per fornire i dati necessari sullo storico, per il calcolo degli indicatori al momento di avvio del testing. Per i limiti temporali di D1 e minori, il volume minimo dello storico scaricato è di un anno.

Pertanto, se si esegue un test su un intervallo 2010.11.01-2010.12.01 (testing per un intervallo di un mese) con un periodo di M15 (ciascuna barra è pari a 15 minuti), il terminale verrà richiesto per lo storico dello strumento finanziario per l'intero anno del 2010. Per il timeframe settimanale, ci verrà richiesta una storia di 100 barre, che è di circa due anni (un anno ha 52 settimane). Per il testing su un timeframe settimanale l'agente chiederà lo storico di 8 anni (12 mesi x 8 anni = 96 mesi).

Se non vi sono le barre necessarie, **la data di inizio del testing verrà automaticamente slittata** dal passato al presente per fornire la necessaria riserva di barre prima del testing.

Durante il testing, il "Market Watch" viene emulato anche, dal quale si possono ottenere informazioni sui simboli.

Per default, all'inizio del testing, c'è solo un simbolo nel "Market Watch" dello Strategy Tester - il simbolo su cui gira il testing. Tutti i simboli necessari sono collegati al "Market Watch" dello Strategy Tester (non al terminale!) automaticamente quando vi si riferiscono.

> Prima di iniziare il testing di un Expert-Advisor multi-valuta, è necessario selezionare i simboli necessari per il testing "Market Watch" del terminale e caricare i dati richiesti. Durante la prima chiamata di un simbolo "straniero", il suo storico verrà automaticamente sincronizzato tra l'agente di testing ed il terminale cliente. Un simbolo "estraneo" è il simbolo diverso da quello su cui testing è in esecuzione.

Il riferimento ai dati di un "altro" simbolo si verificano nei seguenti casi:

- Quando si utilizzano ke funzioni degli indicatori tecnici ed IndicatorCreate() sul simbolo/timeframe;
- La richiesta di dati "Market Watch" per l'altro simbolo:
  1. SeriesInfoInteger
  2. Bars
  3. SymbolSelect
  4. SymbolIsSynchronized
  5. SymbolInfoDouble
  6. SymbolInfoInteger
  7. SymbolInfoString
  8. SymbolInfoTick
  9. SymbolInfoSessionQuote
  10. SymbolInfoSessionTrade
  11. MarketBookAdd
  12. MarketBookGet
- Richiesta della timeseries di un simbolo/timeframe utilizzando le seguenti funzioni:
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh
  6. CopyLow
  7. CopyClose
  8. CopyTickVolume
  9. CopyRealVolume
  10. CopySpread

Al momento della prima chiamata ad un "altro" simbolo, il processo di testing viene interrotto e lo storico viene scaricato per il simbolo/timefreame, dal terminale per l'agente di testing. Allo stesso tempo viene fatta la generazione della sequenza dei tick per questo simbolo.

Una sequenza individuale di tick viene generata per ogni simbolo, secondo il modo selezionato di generazione tick. È anche possibile richiedere lo storico in modo esplicito per i simboli desiderati chiamando SymbolSelect() nell'handler OnInit() - il download dello storico sarà effettuato immediatamente prima del testing dell' Expert Advisor.

Pertanto, non richiede alcuno sforzo supplementare eseguire testing multi-valuta nel terminale client MetaTrader 5 . Basta aprire i grafici dei simboli appropriati del terminale cliente. Lo storico sarà caricato automaticamente dal trade server per tutti i simboli necessari, purché contenga questi dati.

Simulazione del Tempo nello Strategy Tester

Durante il testing, l'ora locale TimeLocal() è sempre uguale all'ora del server TimeTradeServer(). A sua volta, l'orario del server è sempre uguale all'orario corrispondente al tempo GMT - TimeGMT(). In questo modo, tutte queste funzioni visualizzano lo stesso orario durante il testing.

La mancanza di una differenza tra l'orario GMT, locale, server nello Strategy Tester viene fatta deliberatamente in caso non vi è alcuna connessione con il server. I risultati del test devono essere sempre gli stessi, indipendentemente dal fatto che vi sia un collegamento. Le informazioni sull'ora del server non vengono memorizzate localmente, ed sono prese dal server.

Gli Oggetti Grafici nel Testing

Durante il testing/ottimizzazione gli oggetti grafici non vengono tracciati. Così, quando ci si riferisce alle proprietà di un oggetto creato durante il testing/ottimizzazione, un Expert Advisor riceverà i valori zero.

> Questa limitazione non si applica al testing in modalità visuale.

La funzone OnTimer() nello Strategy Tester

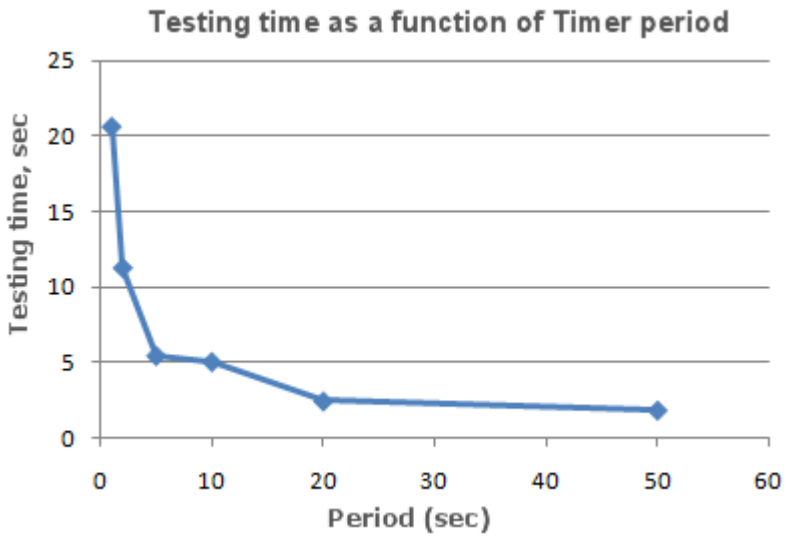
MQL5 offre l'opportunità per l'handling degli eventi timer. La chiamata dell'handler OnTimer() avviene indipendentemente dalla modalità di testing. Ciò significa che se un test è in esecuzione in modalità "Solo prezzi di apertura" per il periodo H4, e l' EA ha un timer impostato ad una chiamata al secondo, allora all' apertura di ogni barra H4, l'handler OnTick() sarà chiamato una sola volta, e l'handler OnTimer() sarà chiamato 14400 volte (3600 secondi * 4 ore). La quantità per cui il tempo di testing dell' EA sarà aumentata dipende dalla logica della EA.

Per verificare la dipendenza del tempo di testing dalla data frequenza del timer, abbiamo creato un EA semplice senza operazioni di trading.

```
//--- parametri di input
input int      timer=1;              // valore timer, sec
input bool     timer_switch_on=true; // timer on
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
```

```
int OnInit()
  {
//--- esegue il timer se timer_switch_on==true
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- ferma il timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Funzione Timer                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// non viene eseguita alcuna azione, il corpo dell'handler è vuoto
  }
//+------------------------------------------------------------------+
```

Misure del tempo di testing sono state prese da diversi valori del parametro timer (periodicità dell'evento Timer). Sui dati ottenuti, abbiamo tracciato un tempo di testing in funzione del periodo di Timer.



Si vede chiaramente che il più piccolo è il parametro di timer, durante l'inizializzazione della funzione EventSetTimer(Timer), più piccolo è il periodo (Period) tra le chiamate dell'handler OnTimer(), e più grande è il tempo T di testing, alle stesse condizioni di altri.

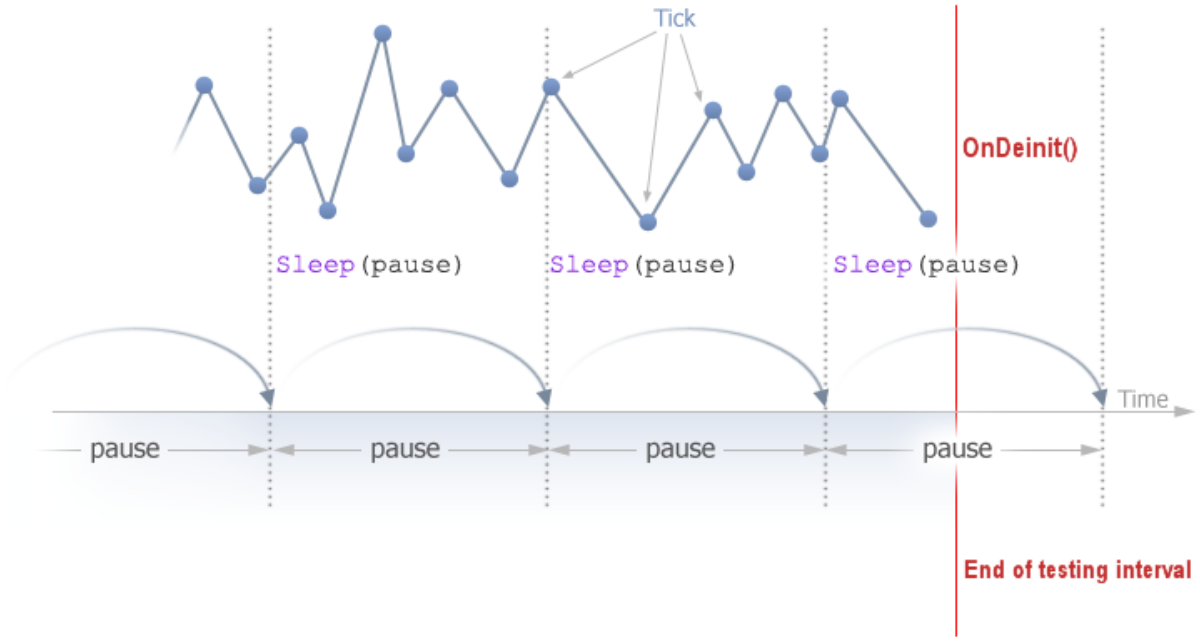
La funzione Sleep() nello Strategy Tester

La funzione Sleep() consente all' EA o script di sospendere l'esecuzione del programma MQL5 per un po', quando si lavora sul grafico. Questo può essere utile per la richiesta di dati, che non sono pronti al momento della richiesta ed è necessario attendere che siano pronti. Un esempio dettagliato di utilizzo della funzione Sleep() possono essere trovati nella sezione Arrangiamento Accesso ai Dati.

Il processo di testing non è attardato dalla chiamata Sleep(). Quando si chiama Sleep(), i ticks generati vengono sono "giocati" entro un termine specificato, che può provocare l'innesco di ordini pendenti, stops, ecc. Dopo una chiamata Sleep(), il tempo simulato nello Strategy Tester aumenta di un intervallo, specificato nel parametro della funzione Sleep.

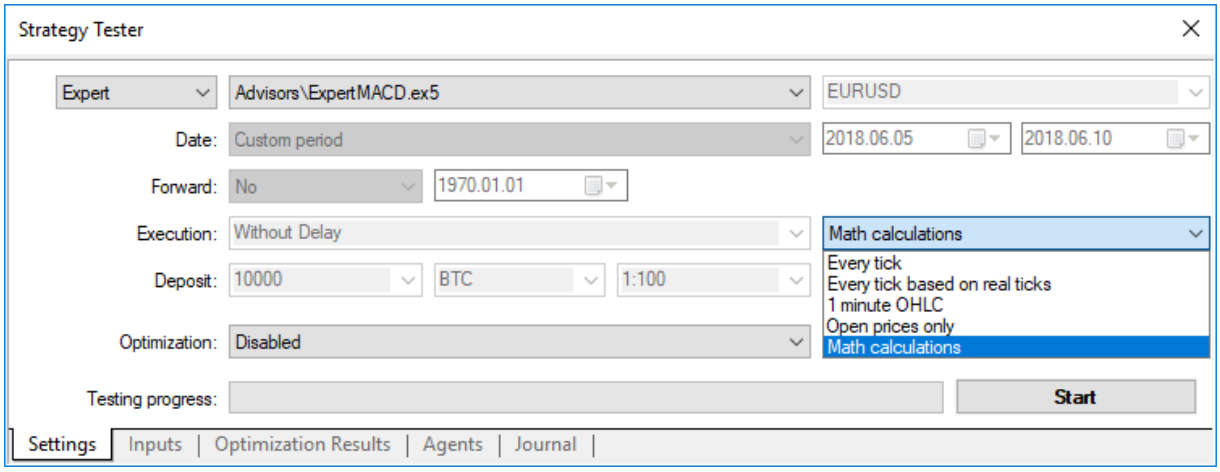
Se, a seguito della esecuzione della funzione Sleep(), l'ora corrente nello Strategy Tester ha superato il periodo di testing, allora si riceverà un errore di "Ciclo si Sleep infinito rilevato durante il testing". Se si riceve questo errore, i risultati del testing non vengono respinti, tutti i calcoli vengono eseguiti nel loro pieno volume (numero di affari, subsidenza, ecc), ed i risultati di questo test vengono trasmessi al terminale.

La funzione Sleep() non funziona in OnDeinit(), dal momento che dopo che viene chiamata, il tempo di testing sarà garantito a superare il range dell' intervallo di testing.



Utilizzo dello Strategy Tester per i Problemi di Ottimizzazione in Calcoli Matematici

Il tester nel terminale MetaTrader 5 può essere utilizzato non solo per testare strategie di trading, ma anche per i calcoli matematici. Per utilizzarlo, è necessario selezionare la modalità "Calcoli Matematici":



In questo caso, solo tre funzioni verranno chiamate: OnInit(), OnTester(), OnDeinit(). In modalità "Calcoli matematici" lo Strategy Tester non genera alcun tick e download dello storico.

Lo Strategy Tester funziona in modalità "Calcoli matematici" anche se si specifica la data di inizio maggiore di quella di chiusura.

Quando si utilizza il tester per risolvere problemi matematici, il caricamento dello storico e la generazione di ticks non si verifica.

Un tipico problema matematico da risolvere nello Strategy Tester MetaTrader 5 - la ricerca di un estremo di una funzione con molte variabili.

Per risolverlo abbiamo bisogno di:

- Il calcolo del valore della funzione deve essere collocato nella funzione OnTester();
- I parametri della funzione devono essere definiti come variabili di input- dell' Expert Advisor;

Compilare l'EA, aprire la finestra "Strategy Tester". Nel campo "Parametri di Input", selezionare le variabili di input richieste, e definire l'insieme dei valori dei parametri specificando i valori di avvio, di arresto e di step per ognuna delle variabili della funzione.

Selezionare il tipo di ottimizzazione - "Algoritmo Completo lento" (ricerca completa dello spazio parametri) o "Algoritmo veloce a base genetica". Per una semplice ricerca di estremo della funzione, è meglio scegliere una ottimizzazione veloce, ma se si desidera calcolare i valori per l'intero set di variabili, allora è meglio usare l'ottimizzazione lenta.

Selezionare la modalità "Calcoli matematici" ed utilizzare il bottone "Start", per eseguire la procedura di ottimizzazione. Notare che durante l'ottimizzazione lo Strategy Tester ricerca i valori massimi della funzione OnTester. Per trovare un minimo locale, restituire l'inverso del valore della funzione calcolata dalla funzione OnTester:

```
return(1/function_value);
```

È necessario controllare che function_value non è uguale a zero, altrimenti si può ottenere un errore critico di divisione per zero.

C'è un altro modo, è più conveniente e non distorce i risultati dell'ottimizzazione, è stato suggerito dai lettori di questo articolo:

```
return(-function_value);
```

Questa opzione non richiede il controllo del function_value_ per essere uguale a zero, e la superficie dei risultati di ottimizzazione in una rappresentazione 3D ha la stessa forma, ma si specchia all'originale.

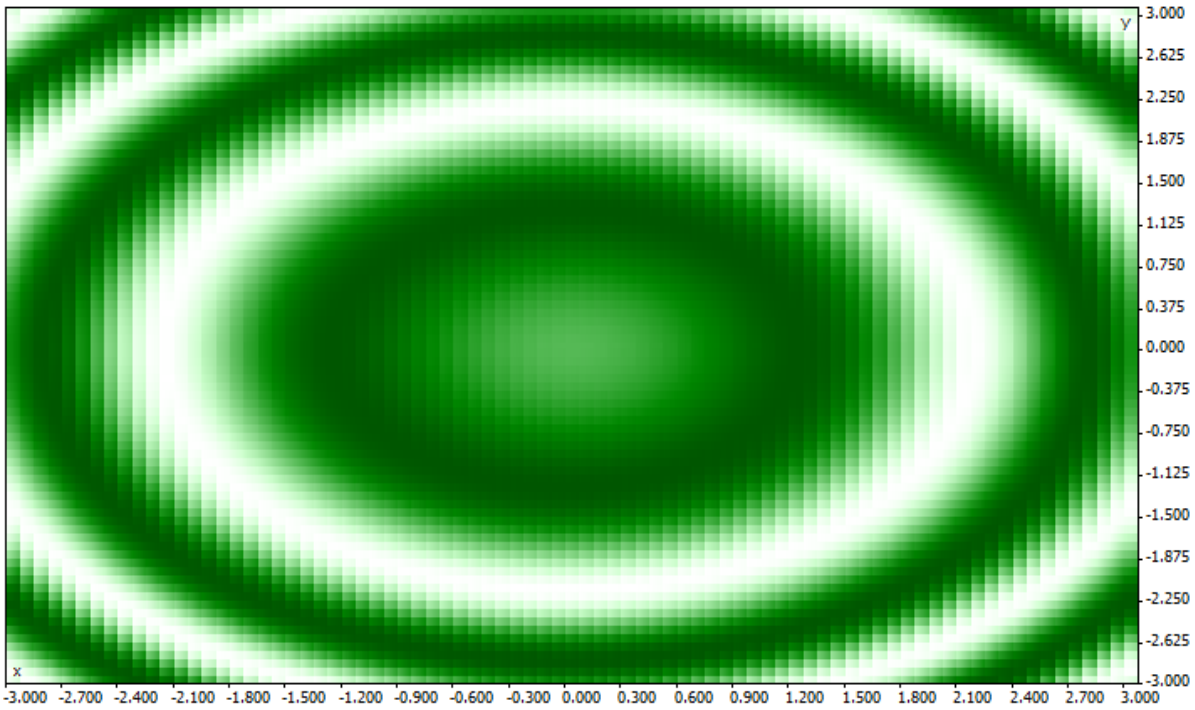
Come esempio, forniamo la funzione sink():

$$sink(x,y) = \sin(x^2 + y^2)$$

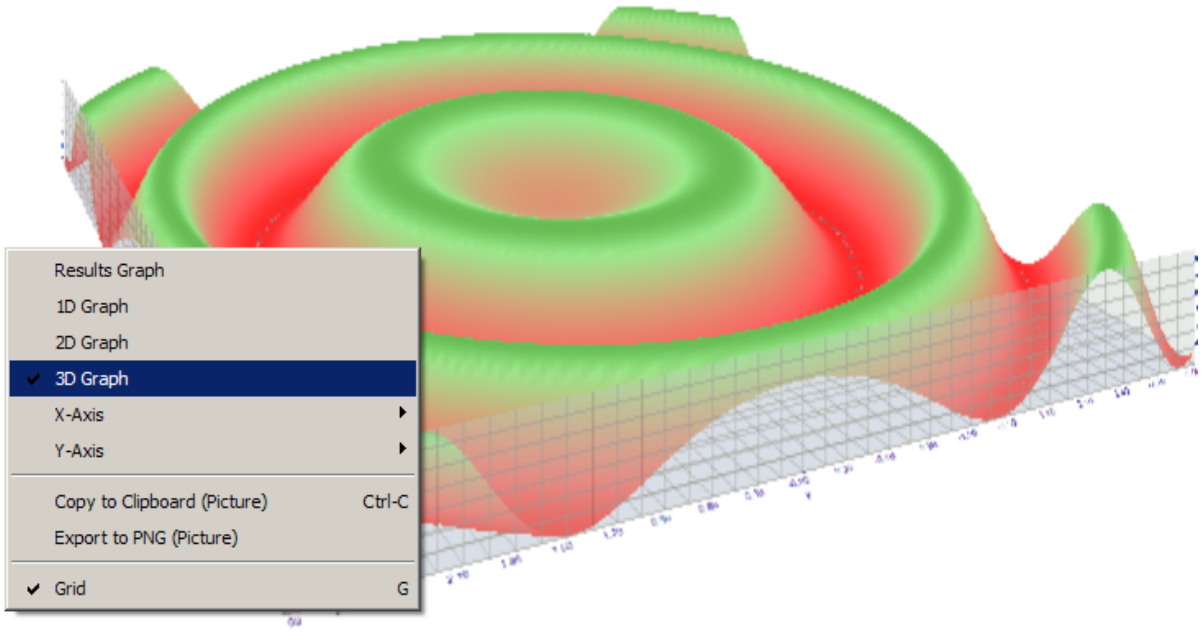
Il codice della EA per trovare l'estremo di questa funzione è inserito nell' OnTester():

```
//+------------------------------------------------------------------------------------+
//|                                                                          Sink.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                                https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parametri di input
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//+------------------------------------------------------------------------------------+
//| Tester function                                                                    |
//+------------------------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------------------------+
```

Eseguie l'ottimizzazione e vede i risultati dell'ottimizzazione sottoforma di un grafico 2D.



Migliore è il valore di una data coppia di parametri (x, y), più è saturo il colore. Come era atteso dalla vista della forma della formula sink(), i suoi valori formano cerchi concentrici con centro in (0,0). Si può vedere nel grafico-3D, che la funzione sink() non ha un estremo unico globale:



La sincronizzazione delle Barre nella modalità "Solo prezzi di apertura"

Il tester nel terminale client MetaTrader 5 ci permette di controllare il cosiddetto "EA multi-valuta". Un EA multi-valuta - è un EA che fa il trade su due o più simboli.

Il testing di strategie, che sono tradate su più simboli, impone alcuni requisiti tecnici complementari sul tester:

- La generazione di ticks per questi simboli;
- Il calcolo dei valori degli indicatori per questi simboli;
- Il calcolo dei requisiti di margine per questi simboli;
- La sincronizzazione di sequenze tick generate per tutti i simboli di trading.

Lo Strategy Tester genera e riproduce una sequenza tick per ogni strumento secondo la modalità trading selezionata. Al tempo stesso, viene aperta una nuova barra per ogni simbolo, indipendentemente da come la barra viene aperta su un altro simbolo. Ciò significa che quando si testa un EA multi-valuta, può verificarsi una situazione (e spesso accade), quando per uno strumento una nuova barra è già aperta, e per l'altro non. Così, durante il testing, tutto avviene proprio come nella realtà.

Questa simulazione autentica dello storico nel tester non causa alcun problema, basta che vengono utilizzate le moddalità di testing "Ogni tick" e "1 minuto OHLC". Per queste modalità, sufficienti ticks vengono generati per una candela, per poter attendere fino a che avviene la sincronizzazione di barre da diversi. Ma come facciamo a testare strategie di un EA multi-valuta in modalità "Solo prezzi di Apertura", se la sincronizzazione delle barre degli strumenti di trading è obbligatoria? In questa modalità, l'EA viene chiamato solo su un tick, che corrisponde al tempo di apertura delle barre.

Noi lo illustrano in un esempio: se stiamo testando un EA su EURUSD, ed una nuova candela oraria è aperta su EURUSD, allora si può facilmente riconoscere questo fatto - nei test in modalità "Solo Prezzi di apertura", l' evento NewTick corrisponde al momento dell'apertura barra sul periodo di testing. Ma non vi è alcuna garanzia che la nuova candela aperta sul simbolo USDJPY, viene utilizzata nella EA.

In circostanze normali, è sufficiente completare il lavoro della funzione OnTick() e verificare la comparsa di una nuova barra sull'emersione di una nuova barra su USDJPY al prossimo tick. Ma, durante il testing in la modalità "Solo prezzi di Apertura", non ci sarà nessun altro tick, e così può sembrare che questa modalità non è adatta per il testing di EA multi-valuta. Ma non è così - non dimenticate che il tester in MetaTrader 5 si comporta proprio come farebbe nella vita reale. È possibile attendere una nuova barra si apre su un altro dei simboli utilizzando la funzione del Sleep() !

Il codice di EA Synchronize_Bars_Use_Sleep.mq5, che mostra un esempio di sincronizzazione di barre in modalità "Solo Prezzi di Apertura":

```
//+------------------------------------------------------------------------------+
//|                                          Synchronize_Bars_Use_Sleep.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parametri di input
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
```

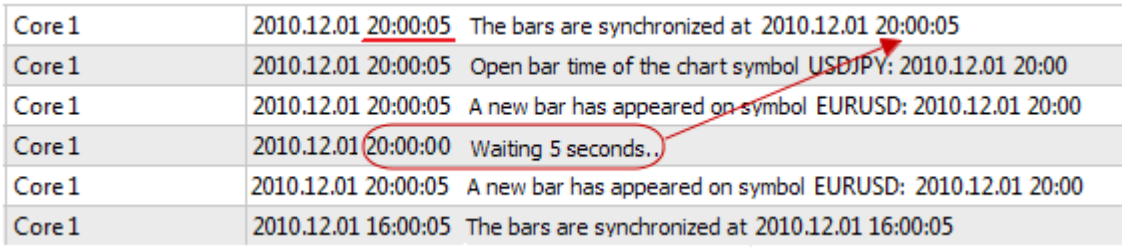
```
  {
//--- controllo il simbolo
   if(_Symbol==other_symbol)
     {
      PrintFormat("Devi specificare l'altro simbolo nei parametri di input o seleziona
      //--- arresto testing forzato
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                    |
//+------------------------------------------------------------------------------+
void OnTick()
  {
//--- variabile statica, utilizzata per il salvataggio del tempo dell'ultima barra
   static datetime last_bar_time=0;
//--- sync flag
   static bool synchonized=false;
//--- se variabile statica non viene inizializzata
   if(last_bar_time==0)
     {
      //--- è la prima chiamata, salva il tempo bar ed esce
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("La variabile last_bar_time è inizializzata con valore %s",TimeToStr
     }
//--- ottiene il tempo di apertura dell'ultima barra del simbolo del grafico
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DAT
//--- se gli orari non sono uguali
   if(curr_time!=last_bar_time)
     {
      //--- salva l'orario di apertura della barra alla variabile statica
      last_bar_time=curr_time;
      //--- non sincronizzato
      synchonized=false;
      //--- stampa messaggio
      PrintFormat("Una nuova barra è apparsa sul simbolo %s at %s",_Symbol,TimeToStrin
     }
//--- prezzo di apertura della barra dell'altro simbolo
   datetime other_time;
//--- loop fino al tempo di apertura di un altro simbolo diventa uguale a curr_time
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SE
     {
      PrintFormat("Aspetta 5 secondi..");
      //--- aspetta 5 secondi e chima SeriesInfoInteger(other_symbol,Period(),SERIES_I
      Sleep(5000);
     }
//--- le barre sono sincronizzate
```

```
   synchonized=true;
   PrintFormat("Orario di aperturara della barra del simbolo del grafico %s: is %s",_S
   PrintFormat("Oerario di apertura della barra del simbolo %s: is %s",other_symbol,Ti
//--- TimeCurrent() non è utile, utilizzare TimeTradeServer()
   Print("Le barre sono sincronizzate a ",TimeToString(TimeTradeServer(),TIME_SECONDS)
  }
//+------------------------------------------------------------------------------+
```

Si noti l'ultima riga dell' EA, che visualizza l'ora corrente quando il fatto della sincronizzazione è stato stabilito:

```
   Print("Le barre sono sincronizzate a ",TimeToString(TimeTradeServer(),TIME_SECONDS)
```

Per visualizzare l'ora corrente abbiamo usato la funzione TimeTradeServer() piuttosto che TimeCurrent(). TimeCurrent() restituisce l'orario dell'ultimo tick, che non cambia dopo l'utilizzo di Sleep(). Esegui l'EA in modalità "Solo Prezzi di Apertura", e vedrai un messaggio riguardo la sincronizzazione delle barre.



Usiamo la funzione TimeTradeServer() al posto di TimeCurrent(), se è necessario per ottenere l'ora corrente del server, e non il tempo di arrivo dell'ultimo tick.

C'è un altro modo per sincronizzare le barre - utilizzando un timer. Un esempio di tale EA è Synchronize_Bars_Use_OnTimer.mq5, che è allegato in questo articolo.

La funzione IndicatorRelease() nella Tester

Dopo aver completato un testing singolo, viene aperto automaticamente un grafico dello strumento, che visualizza le offerte completate e gli indicatori utilizzati per l'EA. Questo aiuta a controllare visivamente i punti di entrata ed uscita, e confrontarli con i valori degli indicatori.

> **Nota:** indicatori, visualizzati sul grafico, che si aprono automaticamente dopo il completamento del test, vengono calcolati nuovamente dopo il completamento del test. Anche se questi indicatori sono stati utilizzati nell' EA testato.

Ma in alcuni casi, il programmatore può decidere di nascondere le informazioni su cui sono stati coinvolti gli indicatori negli algoritmi di trading. Ad esempio, il codice della EA è in affitto o venduto come un file eseguibile, senza la fornitura del codice sorgente. A questo scopo, la funzione IndicatorRelease() è adatta.

Se il terminale imposta un modello con il nome tester.tpl nella directory /profiles/templates del terminale client, allora sarà applicato al grafico aperto. In sua assenza, viene applicato il modello predefinito. (default.tpl).

La funzione IndicatorRelease() è originariamente prevista per rilasciare la porzione di calcolo dell'indicatore, se non è più necessaria. Questo vi permette di salvare sia la memoria che le risorse della CPU, perché ogni tick richiede un calcolo dell'indicatore. Il suo secondo obiettivo - è quello di vietare la visualizzazione di un indicatore sul grafico di testing, dopo una test run singola.

Per proibire la visualizzazione dell'indicatore sul grafico dopo il testing, chiamare IndicatorRelease() con l'handle dell'indicatore nell'handler OnDeinit(). La funzione OnDeinit() viene sempre chiamata dopo il completamento e prima della visualizzazione del grafico di prova.

```
//+------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() completato con successo");
   else Print("IndicatorRelease() ha restituito false. Error code ",GetLastError());
  }
```

Al fine di proibire che si mostri l'indicatore sul grafico, dopo il completamento di un singolo test, utilizzare la funzione IndicatorRelease() nell'handler OnDeinit().

Event Handling nel Tester

La presenza dell' handler OnTick() nell' EA non è obbligatoria per per essere sottoposta a testing su dati storici nel tester MetaTrader 5. È sufficiente che l' EA contenga almeno uno dei seguenti handler-funzione:

- OnTick() - Event handler di arrivo di un nuovo tick;
- OnTrade() - Event handler di trading;
- OnTimer() - Event handler dell'arrivo di un segnale dal timer;
- OnChartEvent() - un handler per gli eventi del client.

Durante il testing in un EA, siamo in grado di gestire gli eventi personalizzati utilizzando la funzione OnChartEvent(), ma negli indicatori, questa funzione non può essere chiamata nel tester. Anche se l'indicatore ha l'event handler OnChartEvent() e questo indicatore è utilizzato nella EA testato, l'indicatore stesso non riceverà alcun evento personalizzato.

Durante il testing, un Indicatore può generare eventi personalizzati utilizzando la funzione EventChartCustom(), e l'EA può elaborare questo evento nell' OnChartEvent().

In aggiunta a questi eventi, eventi speciali associati al processo di testing ed ottimizzazione vengono generati nello strategy tester:

- Tester - questo evento viene generato dopo il completamento del testing dell'Expert Advisor sui dati storici. L'evento Tester viene gestito utilizzando la funzione OnTester(). Questa funzione può essere utilizzata solo quando si fa il testing dell'Expert Advisorè ed destinata principalmente per il calcolo di un valore utilizzato come criterio Custom max per l'ottimizzazione genetica dei parametri di input.
- TesterInit - questo evento viene generato durante l'avvio di ottimizzazione nello strategy tester prima del vero primo passo. L'evento TesterInit viene gestito utilizzando la funzione OnTesterInit().

Durante l'inizio dell' ottimizzazione, un Expert Advisor con questo gestore viene caricato automaticamente in un grafico del terminale separato, con simbolo e periodo specificati nel tester, e riceve l'evento TesterInit. La funzione viene utilizzata per avviare un Expert Advisor prima dell'inizio dell' ottimizzazione per ulteriore elaborazione dei risultati di ottimizzazione.

- TesterPass - questo evento viene generato quando un nuovo frame di dati viene ricevuto. L'evento TesterPass è gestito utilizzando la funzione OnTesterPass(). Un Expert Advisor con questo handler viene caricato automaticamente in un grafico terminale separato con il simbolo/periodo specificati per il testing, e ricevono l'evento TesterPass quando un frame viene ricevuto durante l'ottimizzazione. La funzione viene utilizzata per la gestione dinamica dei risultati di ottimizzazione "sul posto", senza attendere il suo completamento. I frame vengono aggiunti utilizzando la funzione FrameAdd(), che può essere chiamata dopo la fine di un singolo passaggio nell' handler OnTester().
- TesterDeinit - questo evento viene generato dopo la fine della ottimizzazione Expert Advisor nello Strategy Tester. L'evento TesterDeinit è gestito utilizzando la funzione OnTesterDeinit(). Un Expert Advisor con questo handler viene caricato automaticamente in un grafico all'inizio dell' ottimizzazione, e riceve TesterDeinit dopo il suo completamento. La funzione viene utilizzata per la lavorazione finale di tutti i risultati dell'ottimizzazione.

Agenti Testing

Il testing nel terminale MetaTrader 5 viene effettuato utilizzando agenti di test. Agenti locali vengono creati e attivati automaticamente. Il numero predefinito di agenti locali corrisponde al numero di core in un computer.

Ogni agente di testing ha la propria copia delle variabili globali, che non è correlata al terminale client. Il terminale stesso è il dispensatore, che distribuisce i compiti agli agenti locali e remoti. Dopo l'esecuzione di un compito sul testing di un EA, con i parametri specificati, l'agente restituisce i risultati al terminale. Con un singolo test, un solo agente viene utilizzato.

L'agente memorizza lo storico, ricevuto dal terminale, in cartelle separate, dal nome dello strumento, in modo che lo storico per EURUSD è memorizzato in una cartella denominata EURUSD. Inoltre, la storia degli strumenti è separata dalle loro fonti. La struttura per memorizzare lo storico assomiglia al seguente modo:

```
tester_catalog\Agent-IPaddress-Port\bases\name_source\history\symbol_name
```

Ad esempio, lo storico per EURUSD dal server MetaQuotes-Demo può essere memorizzato nella cartella tester_catalog\Agent-127.0.0.1-3000\bases\MetaQuotes-Demo\EURUSD.

Un agente locale, dopo il completamento del testing, entra in una modalità di stand-by, in attesa di una nuova attività per altri 5 minuti, in modo da non perdere tempo per avviare per la chiamata successiva. Solo dopo che il periodo di attesa è finito, l'agente locale si spegne e si decarica dalla memoria della CPU.

In caso di un precoce completamento del testing, da parte dell'utente (il pulsante "Annulla"), nonché con la chiusura del terminale client, tutti gli agenti locali interrompono immediatamente il loro lavoro e vengono decaricati dalla memoria.

Lo Scambio di Dati tra il Terminale e l'Agente

Quando si esegue un test, il terminale client si prepara per inviare all'agente un certo numero di blocchi di parametri:

- I parametri di input per il testing (modalità di simulazione, l'intervallo di test, strumenti, criteri di ottimizzazione, ecc.)
- L'elenco dei simboli selezionati nel "Market Watch"
- La specifica del simbolo di testing (la dimensione del contratto, i margini consentiti dal mercato per l'impostazione di un StopLoss e Takeprofit, ecc)
- L'Expert Advisor da essere testato e i valori dei suoi parametri di input
- Informazioni su file aggiuntivi (librerie, indicatori, file di dati - # property tester_ ...)

tester_indicator	string	Nome di un indicatore personalizzato nel formato "nome_indicatore.ex5". Indicatori che richiedono il testing vengono definiti automaticamente dalla chiamata della funzione iCustom(), se il parametro corrispondente viene impostato tramite una stringa costante. Per tutti gli altri casi (utilizzo della funzione IndicatorCreate() o uso di una stringa non costante nel parametro che imposta il nome dell' indicatore) questa proprietà è obbligatoria
tester_file	string	Nome file per un tester con l'indicazione di estensione, tra virgolette (come una stringa costante). Il file specificato verrà passato al tester. File di input da testare, se ci sono quelli necessari, deve sempre essere specificato.
tester_library	string	Nome libreria con l'estensione, tra virgolette. Una libreria può avere estensione DLL o EX5. Librerie che richiedono testing vengono definite automaticamente. Tuttavia, se una delle librerie viene utilizzata da un indicatore personalizzto, questa proprietà è necessaria

Per ogni blocco di parametri, un'impronta digitale in forma di hash-MD5 viene creata, e viene inviata all'agente. MD5-hash è univoca per ciascun set, il suo volume è molte volte più piccolo della quantità di informazioni sulla quale viene calcolato.

L'agente riceve un hash di blocchi e li confronta con quelli che ha già. Se l'impronta digitale del blocco parametri dato non è presente nell'agente, o l'hash ricevuto è diverso da quello esistente, l'agente richiede questo blocco di parametri. Questo riduce il traffico tra il terminale e l'agente.

Dopo il testing, l'agente restituisce al terminale di tutti i risultati della run, che sono indicati nella scheda "Risultati del Test" e "Risultati Ottimizzazione": il profitto ricevuto, il numero di affari, il coefficiente Sharpe, il risultato della funzione OnTester(), ecc

Durante l'ottimizzazione, le terminale spedisce i compiti di testing agli agenti in piccoli pacchetti, ogni pacchetto contenente diversi compiti (per ogni compito si intende ogni attività di test singolo con una serie di parametri di input). Questo riduce il tempo di scambio tra il terminale e l'agente.

Gli agenti non registrano mai sull'hard disk i files EX5, ottenuti dal terminale (EA, indicatori, librere, ecc) per motivi di sicurezza, di modo che un computer con un agente in esecuzione non può utilizzare i dati inviati. Tutti gli altri file, tra cui DLL, sono registrate nella sandbox. In agenti remoti non si possono testare EA utilizzando DLL.

I risultati dei testing sono sommate dal terminale in una cache speciale dei risultati (i risultati della cache), per un accesso rapido ad essi quando sono necessari. Per ogni insieme di parametri, il

terminale ricerca la i risultati della cache già disponibili dalle run precedenti, al fine di evitare repliche. Se il risultato con un insieme di parametri non viene trovato, all'agente viene dato il compito di eseguire il testing.

Tutto il traffico tra il terminale e l'agente è crittografato.

> I ticks non vengono inviati attraverso la rete, sono generati sugli agenti di testing.

Uso della Cartella Condivisa di tutti i Terminali Client

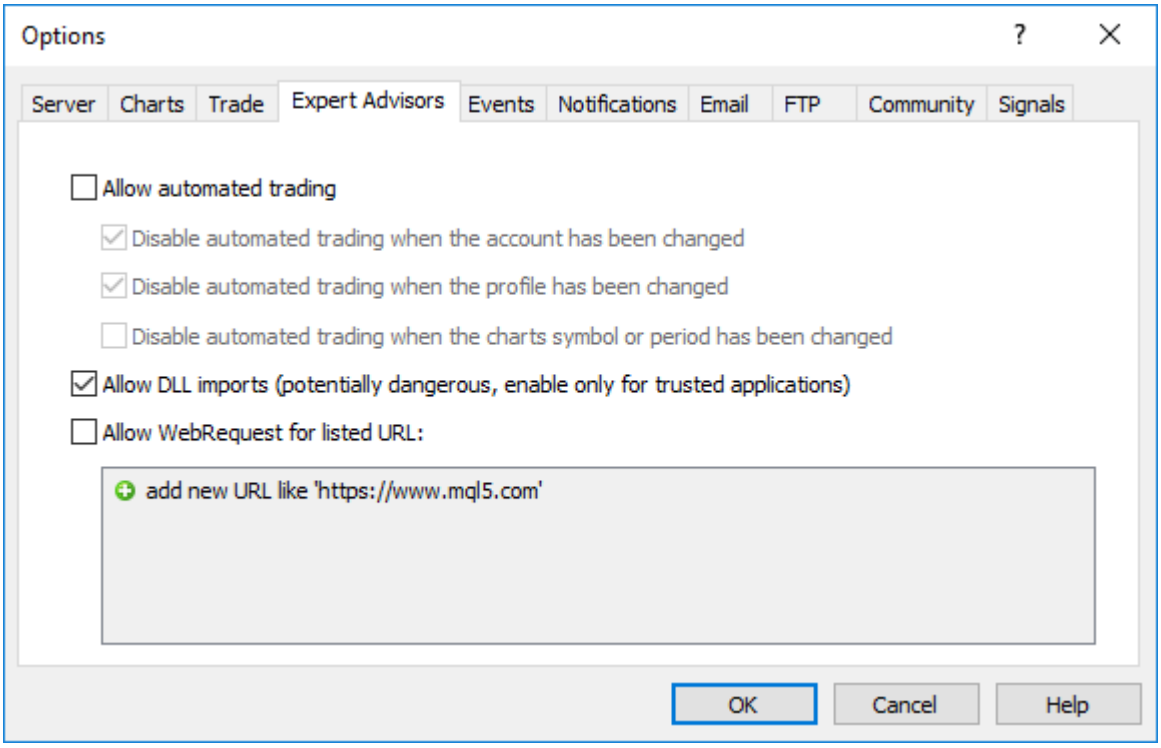
Tutti gli agenti di testing sono isolati gli uni dagli altri e dal terminale client: ogni agente ha una propria cartella in cui sono iscritti i suoi log. Inoltre, tutte le operazioni di file durante il testing dell'agente si verificano nella cartella **agent_name/MQL5/Files**. Tuttavia, siamo in grado di realizzare l'interazione tra gli agenti locali ed il terminale client tramite una cartella condivisa per tutti i terminali client, se durante l'apertura del file si specifica la flag FILE_COMMON:

```
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- la cartella condivisa per tutti i terminali client
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- tira fuori il nome di questa cartella
   PrintFormat("Apre il file nella cartella condivisa dei terminali client %s", commor
//--- apre il file nella cartella condivisa (indicato dal flag FILE_COMMON)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... ulteriori azioni
//---
   return(INIT_SUCCEEDED);
  }
```

Utilizzo di DLLs

Per accelerare l'ottimizzazione è possibile utilizzare non solo agenti locali, ma anche agenti remoti. In questo caso, ci sono alcune limitazioni per agenti remoti. Prima di tutto, gli agenti remoti non mostrano nei loro log i risultati dell'esecuzione della funzione Print(), messaggi sulla apertura e la chiusura delle posizioni. Un minimo di informazioni vengono visualizzate nel registro per evitare che EA erroneamente scritti facciano immondizia nel computer , su cui l'agente di rimozione è in funzione, con i messaggi.

Una seconda limitazione - il divieto di utilizzo di DLL durante il testing degli EA. Le chiamate DLL sono assolutamente vietate su agenti remoti per motivi di sicurezza. In agenti locale, le chiamate DLL negli EA testati sono ammesse solo con l'autorizzazione appropriata "Consenti Importazione DLL".



Nota: Quando si utilizza la riceziona da EA (script, indicatori) che richiedono il permesso di chiamate DLL, si deve essere consapevoli dei rischi che si assumono quando si consente questa opzione nelle impostazioni del terminale. Indipendentemente da come la EA sarà utilizzato - per il testing o per l'esecuzione su un grafico.

Le variabili predefinite

Per ogni programma mql5 eseguibile viene supportata una serie di variabili predefinite, che riflettono lo stato del corrente grafico prezzo, dal momento in cui viene avviato un MQL5-programma (Expert Advisor, script o indicatore personalizzato).

I valori delle variabili predefinite è fissato dal terminale client prima che un programma mql5 venga avviato. Le variabili predefinite sono costanti e non possono essere cambiate da un programma mql5. Come eccezione, vi è una variabile speciale _LastError, che può essere resettata a 0 con la funzione ResetLastError.

Variabile	Valore
_AppliedTo	La variaile _AppliedTo consente di trovare il tipo di dati, utilizzati per il calcolo dell'indicatore
_Digits	Numero di cifre decimali
_Point	Dimensioni del corrente punto simbolo nella valuta di quotazione
_LastError	L'ultimo codice di errore
_Period	Timeframe del grafico corrente
_RandomSeed	Stato corrente del generatore di numeri pseudo-casuali interi
_StopFlag	Flag di arresto programma
_Symbol	Nome Simbolo del grafico corrente
_UninitReason	Codice del motivo di deinizializzazione
_IsX64	La variabile _IsX64 consente di trovare la versione bit del terminale, in cui è in esecuzione un'applicazione MQL5

Le variabili predefinite non possono essere definite in una libreria. Una libreria utilizza tali variabili definite nel programma da cui si chiama questa libreria.

int _AppliedTo

La variaile _AppliedTo consente di trovare il tipo di dati, utilizzati per il calcolo dell'indicatore:

Tipo di dati	Senso	Descrizione dei dati utilizzati per il calcolo dell'indicatore.
—	0	L'indicatore usa la seconda forma di chiamata OnCalculate(): i dati per il calcolo non sono specificati da un determinato buffer o array di dati
Close	1	Prezzi Close
Open	2	Prezzi Open
High	3	Prezzi High
Low	4	Prezzi Low
Prezzo Mediano (HL/2)	5	Prezzo Mediano -- Median Price = (High+Low)/2
Prezzo Tipico (HLC/3)	6	Prezzo Tipico --Typical Price = (High+Low+Close)/3
Prezzo Ponderato (HLCC/4)	7	Prezzo Ponderato -- Weighted price = (Open+High+Low+Close)/4
Dati dell'indicatore precedente	8	Dati dell'indicatore, che è stato lanciato sul chart prima di questo indicatore
Dati del primo indicatore	9	Dati dell'indicatore, che è stato lanciato per primo sul chart
Handle dell'indicatore	10+	Dati dell'indicatore, che è stato passato alla funzione iCustom() usando l'handle dell'indicatore. Il valore _AppliedTo contiene l'handle dell'indicatore

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Inizializzazione Indicatore Custom                      |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura dei buffer degli indicatori
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// Ottenere il tipo di dati utilizzati per il calcolo dell'indicatore
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
//---
   return(INIT_SUCCEEDED);
  }
```

```
//+------------------------------------------------------------------+
//| Descrizione dei dati utilizzati per il calcolo dell'indicatore   |
//+------------------------------------------------------------------+
string getIndicatorDataDescription(int data_id)
  {
   string descr="";
   switch(data_id)
     {
      case(0):descr="È il primo tipo di OnCalculate() - nessun buffer di dati";
         break;
      case(1):descr="l' indicatore calcola sul prezzo Close";
         break;
      case(2):descr="l' indicatore calcola sul prezzo Open";
         break;
      case(3):descr="l' indicatore calcola sul prezzo High";
         break;
      case(4):descr="l' indicatore calcola sul prezzo Low";
         break;
      case(5):descr="l' indicatore calcola sul prezzo Median Price (HL/2)";
         break;
      case(6):descr="l' indicatore calcola sul prezzo Typical Price (HLC/3)";
         break;
      case(7):descr="l' indicatore calcola sul prezzo Weighted Price (HLCC/4)";
         break;
      case(8):descr="L'indicatore calcola i dati dell'indicatore precedente";
         break;
      case(9):descr="L'indicatore calcola i dati del Primo Indicatore";
         break;
      default: descr="L'indicatore calcola i dati dell'indicatore con handle="+string
         break;
     }
//---
   return descr;
  }
```

Guarda anche

ENUM_APPLIED_PRICE

int _Digits

La variabile _Digits memorizza numeri di cifre dopo il punto decimale, che definisce la precisione prezzo del simbolo del grafico corrente.

Si può anche usare la funzione Digits().

double _Point

La variabile _Point contiene la grandezza in punti del simbolo corrente nella valuta di quotazione.

Si può anche usare la funzione Point().

int _LastError

La variabile _LastError contiene il codice dell'ultimo errore, che si è verificato nel corso del programma mql5. Il suo valore può essere resettato a zero ResetLastError().

Per ottenere il codice dell'ultimo errore, si può anche usare la funzione GetLastError().

ENUM_TIMEFRAMES _Period

La variabile _Period contiene il valore del periodo del grafico corrente.

Inoltre è possibile utilizzare la funzione Period().

Vedi anche

PeriodSeconds, Timeframes del Grafico, Data ed Ora, Visibilità degli Oggetti

_RandomSeed

Variabile per la memorizzazione dello stato attuale quando si generano numeri pseudo-casuali interi. _RandomSeed cambia il suo valore quando si chiama MathRand(). Utilizzare MathSrand() per impostare la condizione richiesta iniziale.

x il numero casuale ricevuto da MathRand() viene calcolato nel seguente modo ad ogni chiamata:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

Vedi anche

MathRand(), MathSrand(), Tipi interi

bool _StopFlag

La variabile _StopFlag contiene la flag di stop del programma mql5. Quando il terminale client sta cercando di fermare il programma, imposta la variabile _StopFlag su true.

Per controllare lo stato del _StopFlag si può anche usare la funzione IsStopped().

string _Symbol

La variabile _Symbol contiene il nome del simbolo del grafico corrente.

Si può anche usare la funzione Symbol().

int _UninitReason

La variabile _UninitReason contiene il codice del motivo di deinizializzazione del programma.

Solitamente, questo codice è ottenuto dalla funzione UninitializeReason().

int _IsX64

La variabile _IsX64 consente di trovare la versione bit del terminale, in cui è in esecuzione un'applicazione MQL5: _IsX64=0 per il terminale a 32 bit e _IsX64!= 0 per il terminale a 64 bit.

Inoltre può essere utilizzata la funzione TerminalInfoInteger(TERMINAL_X64).

Esempio:

```
// Controllo del terminale, in cui è in esecuzione il programma
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("Program ",__FILE__," sta girando sul terminale 64-bit");
   else
      Print("Program ",__FILE__," sta girando sul terminale 32-bit");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

Guarda anche

MQLInfoInteger, Funzioni di Importazione (#import)

Common Functions

Le funzioni a scopo-generico non incluce in nessun gruppo specializzato, sono qui elencate.

Funzione	Azione
Alert	Mostra un messaggio in una finestra separata.
CheckPointer	Restituisce il tipo di pontatore oggetto
Comment	Manda in output un commento, nell'angolo superiore sinistro del grafico
CryptEncode	Transforms the data from array with the specified method
CryptDecode	Performs the inverse transformation of the data from array
DebugBreak	Breakpoint del programma nel debugging
ExpertRemove	Ferma l' Expert Advisor e lo decarica dal grafico
GetPointer	Restituisce il pointer dell'oggetto
GetTickCount	Restituisce il numero di millisecondi che sono passati dal momento in cui il sistema è partito
GetTickCount64	Restituisce il numero di millisecondi che sono passati dal momento in cui il sistema è partito
GetMicrosecondCount	Restituisce il numero di microsecondi trascorsi dall'inizio del programma-MQL5
MessageBox	Crea e mostra un box messaggi e lo gestisce
PeriodSeconds	Restituisce il numero di secondi nel periodo
PlaySound	Riproduce un file sonoro
Print	Mostra un messaggio nel log
PrintFormat	Formatta e stampa il set di simboli e valori in un file di log secondo il presente formato
ResetLastError	Imposta a zero il valore di una variabile predeterminata _LastError
ResourceCreate	Crea una risorsa immagine basata su un set di dati
ResourceFree	Elimina le risorse create dinamicamente (liberando la memoria allocata per esse)
ResourceReadImage	Legge i dati dalla risorsa grafica creata dalla funzione ResourceCreate() o salvati nel file EX5 durante la compilazione
ResourceSav	Salva una risorsa nel file specificato
SetUserError	Imposta la variabile predefinita _LastError nel valore pari a ERR_USER_ERROR_FIRST + user_error

Funzione	Azione
SetReturnError	Imposta il codice che restituisce il processo del terminale quando si completa l'operazione
Sleep	Sospende l'esecuzione del corrente Expert Advisor o Script in un intervallo specificato
TerminalClose	Comanda al terminale di completare l'operazione
TesterHideIndicators	Imposta la modalità di visualizzazione/occultamento degli indicatori utilizzati in un EA
TesterStatistics	Restituisce il valore di una statistica specificata calcolata in base ai risultati di testing
TesterStop	Fornisce il comando di completamento dell'operazione del programma quando si fa il testing
TesterDeposit	La funzione speciale che emula i fondi depositati, durante un test. Può essere utilizzato in alcuni sistemi di gestione del denaro
TesterWithdrawal	La funzione speciale per emulare il l'operazione di prelievo del denaro nel processo di testing
TranslateKey	Returns a Unicode character by a virtual key code
ZeroMemory	Resetta la variabile passata ad esso per riferimento. La variabile può essere di qualsiasi tipo, eccetto per le classi e strutture che hanno costruttori.

Alert

Visualizza un messaggio in una finestra separata.

```
void  Alert(
   argument,      // primo valore
   ...            // altri valori
   );
```

Parametri

argomento

[in] Tutti i valori separati da virgole. Per dividere l'output delle informazioni in più righe è possibile utilizzare il carattere di avanzamento riga "\n" o "\r\n". Il numero di parametri non può superare 64.

Valore restituito

Nessun valore restituito.

Nota

Gli array non possono essere passati alla funzione Alert(). Arrays should be output elementwise I dati di tipo double sono output con 8 cifre dopo la virgola, i dati di tipo float vengono visualizzati con 5 cifre dopo la virgola. Per emettere in output i numeri reali con una precisione diversa o in un formato scientifico, utilizzare la funzione DoubleToString().

I dati di tipo bool è sono dati in output come stringhe "true" o "false". Le date sono emesse in output come AAAA.MM.GG. HH:MI:SS. Per visualizzare una data in un altro formato utilizzare la funzione TimeToString(). I dati di tipo colore vengono emessi in output sia come stringhe R,G,B o come un nome di colore, se il colore è presente in un set di colori.

Alert() function does not work in the Strategy Tester.

CheckPointer

La funzione restituisce il tipo dell'oggetto pointer.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // puntatore oggetto
   );
```

Parametri

anyobject

[in] Object pointer.

Return value

Restituisce un valore dall'enumerazione ENUM_POINTER_TYPE.

Nota

Un tentativo di chiamare un puntatore scorretto risulta nella terminazione critica del programma. Ecco perché è necessario chiamare la funzione CheckPointer prima di usare un puntatore. Un puntatore può essere errato nei seguenti casi:

- il puntatore è uguale a NULL;
- l'oggetto è stato eliminato con l'operatore delete.

Questa funzione può essere utilizzata per verificare la validità del puntatore. Un valore non-zero garantisce che il puntatore può essere utilizzato per l'accesso.

Per convalidare rapidamente il puntatore, puoi anche utilizzare l'operatore "!" (esempio) che lo verifica tramite una chiamata implicita della funzione CheckPointer.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Elimina la lista eliminando i suoi elementi                                  |
//+------------------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- servizio puntatore per lavorare nel loop
   CItem* item;
//--- passa attraverso il loop e tenta di eliminare i puntatori dinamici
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Oggetto dinamico ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("L' oggetto non-dinamico ",item.Identifier()," non può essere elimina
     }
```

```
//---
  }
```

Vedi anche

Object Pointers, Checking the Object Pointer, Object Delete Operator delete

Comment

Questa funzione emette in output un commento definito da un utente in alto a sinistra di un grafico.

```
void  Comment(
   argument,      // primo valore
   ...            // prossimo valore
   );
```

Parametri

...

[in] Tutti i valori, separati da virgole. Per delimitare i dati in output in diverse righe, viene usato un simbolo di interruzione di riga "\n" o "\r\n". Il numero di parametri non può superare 64. La lunghezza totale del commento di input (compresi i simboli invisibili) non può superare i 2045 caratteri (i simboli in eccesso verranno tagliati durante l'output).

Valore restituito

Nessun valore di ritorno

Nota

Gli array non possono essere passati alla funzione Comment(). Gli array devono essere riempiti elemento per elemento.

I dati di tipo double vengono emessi in output con la precisione di un massimo di 16 cifre dopo la virgola decimale, e possono essere inviati in output sia in formato tradizionale che in formato scientifico, a seconda di quale notazione sarà più compatta. I dati di tipo float sono output con 5 cifre dopo la virgola decimale. Per emettere in output i numeri reali con un altra precisione o in un formato predefinito, utilizzare la funzione DoubleToString().

I dati di tipo bool vengono emessi in output come stringhe "true" o "false". Le date vengono visualizzati come AAAA.MM.GG. HH:MI:SS. Per visualizzare le date in un altro formato, utilizzare la funzione TimeToString(). Dati del tipo color vengono emessi in output sia come stringhe R,G,B o come nome di colore, se questo colore è presente nel set di colori.

La funzione Comment() non funziona durante l'ottimizzazione nel Tester di strategia.

Esempio:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- Valori di uscita in tre righe
   Comment(StringFormat("Show prices\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,Spread)
  }
```

Vedi anche

ChartSetString, ChartGetString

CryptEncode

Transforms the data from array with the specified method.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,         // method
   const uchar&        data[],         // source array
   const uchar&        key[],          // key
   uchar&              result[]        // destination array
   );
```

Parameters

method

[in] Data transformation method. Can be one of the values of ENUM_CRYPT_METHOD enumeration.

data[]

[in] Source array.

key[]

[in] Key array.

result[]

[out] Destination array.

Returned value

Amount of bytes in the destination array or 0 in case of error. To obtain information about the error call the GetLastError() function.

Example:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- check
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- transform to HEX string
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

```
  {
   string text="The quick brown fox jumps over the lazy dog";
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- prepare key
   StringToCharArray(keystr,key);
//--- copy text to source array src[]
   StringToCharArray(text,src);
//--- print initial data
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
//--- encrypt src[] with DES 56-bit key in key[]
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- check error
   if(res>0)
     {
      //--- print encrypted data
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- decode dst[] to src[]
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- check error
      if(res>0)
        {
         //--- print decoded data
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("Error in CryptDecode. Error code=",GetLastError());
     }
   else
      Print("Error in CryptEncode. Error code=",GetLastError());
  }
```

See also

Array Functions, CryptDecode()

CryptDecode

Performs the inverse transformation of the data from array, tranformed by CryptEncode().

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,        // method
   const uchar&        data[],        // source array
   const uchar&        key[],         // key
   uchar&              result[]       // destination array
   );
```

Parameters

method

[in] Data transformation method. Can be one of the values of ENUM_CRYPT_METHOD enumeration.

data[]

[in] Source array.

key[]

[in] Key array.

result[]

[out] Destination array.

Returned value

Amount of bytes in the destination array or 0 in case of error. To obtain information about the error call the GetLastError() function.

See also

Array Functions, CryptEncode()

DebugBreak

Si tratta di un punto di interruzione nel programma di debug.

```
void  DebugBreak();
```

Valore restituito

Nessun valore restituito.

Nota

L'esecuzione di un programma MQL5 viene interrotta solo se un programma viene avviato in modalità di debug. La funzione può essere utilizzata per visualizzare i valori delle variabili e/o per un'ulteriore esecuzione passo-passo.

ExpertRemove

La funzione ferma un Expert Advisor e lo decarica da un grafico.

```
void  ExpertRemove();
```

Valore restituito

Nessun valore restituito.

Nota

L'Expert Advisor non viene fermato immediatamente appena si chiama ExpertRemove(); viene impostato solo un flag per fermare l'operatività dell' EA . Vale a dire, che ogni evento successivo non verrà elaborato, verr chiamato OnDeinit() e l'Expert Advisor verrà decaricato e rimosso dal grafico.

Chiamare ExpertRemove() nel tester di strategia all'interno dell handler OnInit() annulla i test sul set corrente di parametri. Tale completamento è considerato un errore di inizializzazione.

Quando si chiama ExpertRemove() nel tester della strategia dopo inizializzazione riuscita di un EA, un test è completato normalmente con la chiamata di OnDeinit() e OnTester(). In questo caso, vengono ottenuti i valori delle intere statistiche di trading e criterio di ottimizzazione.

Esempio:

```
//+------------------------------------------------------------------+
//|                                                Test_ExpertRemove.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20;// number of ticks before EA unload
//+------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," reason code = ",reason);
//--- "clear" comment
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| Funzione tick dell'Expert                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
   static int tick_counter=0;
//---
```

```
   tick_counter++;
   Comment("\nPrima di decaricare l'expert advisor ",__FILE__," left",
           (ticks_to_close-tick_counter)," ticks");
//--- prima
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," l'expert advisor verrà decaricato");
     }
   Print("tick_counter =",tick_counter);
//---
  }
//+------------------------------------------------------------------------------+
```

Vedi anche

Funzionamento programmi, Eventi terminale client

GetPointer

La funzione restituisce l'oggetto pointer.

```
void*  GetPointer(
   any_class anyobject      // oggetto di qualsiasi classe
   );
```

Parametri

anyobject

[in] Oggetto di qualsiasi classe.

Valore restituito

La funzione restituisce il puntatore all'oggetto.

Nota

Solo gli oggetti della classe hanno puntatori. Le istanze delle strutture e variabili di tipo semplice non possono avere puntatori. L'oggetto della classe creato non usando l'operatore new(), ma, ad esempio, creato automaticamente nell'array di oggetti, ha ancora un puntatore. Ma questo puntatore sarà di tipo automatico POINTER_AUTOMATIC, pertanto l'operatore delete() non può essere applicato. A parte questo, il puntatore tipo non differisce da puntatori dinamici di tipo POINTER_DYNAMIC.

Poiché le variabili di tipo strutture e di tipo semplice non sono puntatori, è vietato applicare la funzione GetPointer() per esse. E' anche vietato passare il puntatore come argomento di funzione. In tutti questi casi il compilatore notificherà un errore.

Un tentativo di chiamare un puntatore errato provoca la terminazione critica del programma. Ecco perché la funzione CheckPointer() deve essere chiamata prima di utilizzare un puntatore. Un puntatore può essere corretto nei seguenti casi:

- il puntatore è uguale a NULL;
- l'oggetto è stato eliminato con l'operatore delete.

Questa funzione può essere utilizzata per verificare la validità di un puntatore. Un valore non-zero garantisce, che il puntatore può essere utilizzato per accedere.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                    Check_GetPointer.mq5 |
//|                                Copyright 2009, MetaQuotes Software Corp. |
//|                                                    https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------------------+
//| Classe che implementa gli elementi della lista                              |
```

```
//+------------------------------------------------------------------------------+
class CItem
  {
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("Distruttore di ",m_id,
                                     (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                     "dynamic":"non-dynamic"); }
   void              Initialize(int id,string comm) { m_id=id; m_comment=comm; }
   void              PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
   int               Identifier() { return(m_id); }
   CItem*            Next() {return(m_next); }
   void              Next(CItem *item) { m_next=item; }
  };
//+------------------------------------------------------------------------------+
//| Più semplice classe della lista                                              |
//+------------------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------------------+
//| Inserisce gli elementi della lista all'inizio                                |
//+------------------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Elimina la lista eliminando gli elementi                                     |
//+------------------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- puntatore di servizio per lavorare in un loop
   CItem* item;
//--- passa attraverso il loop e tenta di eliminare i puntatori dinamici
```

```
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamyc object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("L' oggetto non-dinamico ",item.Identifier()," non può essere elimina
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- crea ed aggiunge nella lista un puntatore oggetto dinamico
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- aggiunge puntatori automatici nella lista
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automatic");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- aggiunge uno o più oggetti puntatori dinamici all'inizio della lista
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- elimina tutti gli elementi della lista
   list.Destroy();
//--- tutti gli elementi della lista verranno eliminati dopo che lo script è andato
```

```
//--- vedere la scheda Experts nel terminale
  }
```

Vedi anche

Object Pointers, Checking the Object Pointer, Object Delete Operator delete

GetTickCount

La funzione GetTickCount() restituisce il numero di millisecondi trascorsi dall'inizio l'avvio del sistema.

```
uint  GetTickCount();
```

Valore restituito

Valore di tipo uint.

Nota

Il contatore è limitato dalle restrizioni del timer di sistema. L'orario viene memorizzato come numero intero senza segno, così viene sovrariempito ogni 49,7 giorni se il computer lavora ininterrottamente.

Esempio:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------------------------+
//| Script per misurare il tempo di calcolo di 40 numeri di Fibonacci                  |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Ricordate il valore iniziale
   uint start=GetTickCount();
//--- Una variabile per ottenere il numero successivo della serie di Fibonacci
   long fib=0;
//--- In loop calcola la quantità specificata di numeri dalla serie di Fibonacci
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- Prende il tempo trascorso in millisecondi
   uint time=GetTickCount()-start;
//--- Manda in output un messaggio al journal Experts
   PrintFormat("Calcola i primi %d numeri di Fibonacci prendendo %d millisecondo",MAX_
//--- Script completato
   return;
  }
//+------------------------------------------------------------------------------------+
//| Funzione per ottenere il numero di Fibonacci per il suo numero di serie            |
//+------------------------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- Il primo membro della serie di Fibonacci
   if(n<2) return(1);
//--- Tutti gli altri membri vengono calcolati con la seguente formula
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Vedi anche

Data ed Ora, EventSetMillisecondTimer, GetTickCount64, GetMicrosecondCount

GetTickCount64

La funzione GetTickCount64() restituisce il numero di millisecondi trascorsi dall'avvio del sistema.

```
ulong  GetTickCount64();
```

Valore di ritorno

Un valore di tipo ulong.

Nota

Il contatore è limitato alla precisione del timer di sistema, che di solito restituisce un risultato con una precisione di 10-16 millisecondi. Diversamente da GetTickCount, che è di tipo uint e il cui contenuto overflowa ogni 49.7 giorni in caso di funzionamento continuo del computer, GetTickCount64() può essere utilizzato per un tempo di funzionamento illimitato del computer e non è soggetto ad overflow.

Guarda anche

Data ed Ora, EventSetMillisecondTimer, GetTickCount, GetMicrosecondCount

GetMicrosecondCount

La funzione GetMicrosecondCount() restituisce il numero di microsecondi trascorsi dall'inizio del programma-MQL5.

```
ulong  GetMicrosecondCount();
```

Valore restituito

Valore di tipo ulong.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Test                                                    |
//+------------------------------------------------------------------+
void Test()
  {
   int    res_int=0;
   double res_double=0;
//---
   for(int i=0;i<10000;i++)
     {
      res_int+=i*i;
      res_int++;
      res_double+=i*i;
      res_double++;
     }
  }
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- numero di misurazioni
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- seleziona il tipo di misurazione
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- esecuzione codice
         Test();
         //--- aggiungi il risultato della misurazione (a seconda del tipo)
```

```
         if(n==0) ui_res+=GetTickCount()-ui;
         else     ul_res+=GetMicrosecondCount()-ul;
        }
      //--- calcola il tempo minimo e massimo per entrambe le misurazioni
      if(ui_min>ui_res) ui_min=ui_res;
      if(ui_max<ui_res) ui_max=ui_res;
      if(ul_min>ul_res) ul_min=ul_res;
      if(ul_max<ul_res) ul_max=ul_res;
     }
//---
   Print("GetTickCount error(msec): ",ui_max-ui_min);
   Print("GetMicrosecondCount error(msec): ",DoubleToString((ul_max-ul_min)/1000.0,2))
  }
```

Vedi anche

Data ed Ora, GetTickCount, GetTickCount64

MessageBox

Crea e mostra una finestra di messaggio e lo gestisce. Una finestra di messaggio contiene un messaggio e l'intestazione, qualsiasi combinazione di segni predefiniti e pulsanti di comando.

```
int  MessageBox(
   string  text,              // testo del messaggio
   string  caption=NULL,      // box header
   int     flags=0            // definisce il set di bottoni nel box
   );
```

Parametri

text

[in] Testo, contenente il messaggio da dare in output.

caption=NULL

[in] Testo opzionale da essere visualizzato nell'intestazione del box . Se il parametro è vuoto, il nome Expert Advisor è indicato nell'intestazione del box.

flags=0

[in] flags opzionale che definisce l'aspetto ed il comportamento del box messaggio. Flags can be a combination of a special group of flags.

Valore restituito

Se la funzione è eseguita con successo, il valore restituito è uno dei valori dei codici di ritorno di MessageBox().

Nota

La funzione non può essere chiamata da indicatori personalizzati, in quanto gli indicatori vengono eseguiti nel thread di interfaccia e non dovrebbero rallentarla.

MessageBox() function does not work in the Strategy Tester.

PeriodSeconds

Questa funzione restituisce il numero di secondi in un periodo.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // periodo del grafico
   );
```

Parametri

period=PERIOD_CURRENT

[in] Valore di un periodo grafico dall'enumerazione ENUM_TIMEFRAMES. Se il parametro non viene specificato, restituisce il numero di secondi del periodo del grafico corrente, in cui il programma viene eseguito.

Valore restituito

Numero di secondi in un determinato periodo.

Vedi anche

_Period, Timeframes del Grafico, Data ed Ora, Visibilità degli Oggetti

PlaySound

Riproduce un file sonoro

```
bool  PlaySound(
   string  filename      // nome del file
   );
```

Parametri

filename

[in] Percorso del file audio. Se filename=NULL, il playback viene fermato.

Valore restituito

true - se il file viene trovato, in caso contrario - false.

Nota

Il file deve essere posizionato in terminal_directory\Sounds o la sua sub-directory. Solo i file WAV vengono riprodotti.

La chiamata di PlaySound() con il parametro NULL ferma il playback.

PlaySound() function does not work in the Strategy Tester.

Vedi anche

Risorse

Print

Inserisce un messaggio nel registro Expert Advisor. I parametri possono essere di qualsiasi tipo.

```
void  Print(
   argument,      // primo valore
   ...            // prossimo valore
   );
```

Parametri

...

[in] Tutti i valori separati da virgole. Il numero di parametri non può superare 64.

Nota

Gli array non possono essere passati alla funzione Print(). Gli array devono essere inseriti elemento per elemento.

I dati di tipo double vengono indicati con la precisione di un massimo di 16 cifre dopo la virgola decimale, e possono essere emessi in output sia in formato scientifico che tradizionale, a seconda di quale entry sarà piu compatta. I dati di tipo float sono output con 5 cifre dopo la virgola decimale. Per emettere in output i numeri reali con un altra precisione o in un formato predefinito, utilizzare la funzione PrintFormat().

I dati di tipo bool vengono emessi come righe "true" o "false". Le date vengono visualizzati come AAAA.MM.GG. HH:MI:SS. Per visualizzare i dati in un altro formato, utilizzare TimeToString(). Dati del tipo di colore vengono restituiti sia come righe R,G,B o come nome di colore, se questo colore è presente nel set di colori.

La funzione Print() non funziona durante l'ottimizzazione nel Tester di strategia.

Esempio:

```
voidOnStart()
  {
//--- Da in output DBL_MAX usando Print(), questo è equivalenta a PrintFormat(%%.16G,I
   Print("---- come appaere DBL_MAX -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//--- Ora da in output un numero DBL_MAX usando PrintFormat()
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Output all' Experts journal
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- Vedi come il float è dato in output
   float c=(float)M_PI; // Dobbiamo esplicitamente castarlo al tipo target
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- Mostra cosa può accadere con le operazioni aritmetiche con tipi reali
   double a=7,b=200;
   Print("---- Prima delle operazioni aritmetiche");
```

```
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Divide a per b (7/200)
   a=a/b;
//--- Ora emula il ripristino di un valore della variabile b
   b=7.0/a; // It is expected that b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - but it diffe
//--- Da in output il nuovo valore calcolato di b
   Print("----- Dopo le operazioni aritmetiche");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Output all' Experts journal
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 (vedi che b non è più uguale a 200

//--- Crea valore espilon molto piccolo=1E-013
   double epsilon=1e-13;
   Print("---- Crea un valore molto piccolo");
   Print("epsilon=",epsilon); // Ottiene epsilon=1E-013
//--- Ora sottrae epsilon da B e di nuovo da in output il valore al journal Experts
   b=b-epsilon;
//--- Usa due modi
   Print("---- Dopo aver sottratto epsilon dalla variabile b");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Output all' Experts journal
// Print(b)=199.9999999999999  (ora il valore di b sopo aver sottratto epsilon non puo
// Print(DoubleToString(b,16))=199.9999999999998578
//    (ora il valore di b dopo aver sottratto epsilon non può essere arrotondato a 200
  }
```

Vedi anche

DoubleToString, StringFormat

PrintFormat

Formatta ed introduce un set di simboli e valori nel registro Expert Advisor secondo un formato prestabilito.

```
void  PrintFormat(
   string format_string,   // formato stringa
   ...                     // valori dei tipi semplici
   );
```

Parametri

format_string

[in] Un formato stringa è composto di simboli semplici, e se il formato stringa è seguito da argomenti, contiene anche le specifiche di formato.

...

[in] Tutti i valori di tipi semplici separati da virgole. Il numero di parametri non può superare i 64 compreso il formato stringa.

Valore restituito

String.

Nota

La funzione PrintFormat() non funziona durante l'ottimizzazione nel Tester di strategia.

Il numero, l'ordine e il tipo di parametri devono corrispondere esattamente al set di qualificatori, altrimenti il risultato di stampa è indefinito. Invece di PrintFormat() è possibile utilizzare printf().

Se il formato stringa è seguito dai parametri, questa stringa deve contenere le specifiche di formato che indicano il formato di output di questi parametri. Specification of format always starts with the percent sign (%).

Un formato stringa viene letto da sinistra a destra. Quando viene soddisfatta la prima specifica di formato (se presente), il valore del primo parametro dopo il formato stringa viene trasformato ed emesso in output secondo la specifica preset. La specifica del secondo formato chiama la trasformazione e l'output del secondo parametro, e così via fino alla fine del formato stringa.

La specifica del formato ha la seguente forma:

%[flags][width][.precision][{h | l | ll | I32 | I64}]type

Ogni campo della specifica di formato è un simbolo semplice, o un numero che denota una semplice opzione formato. La più semplice specifica di formato contiene solo il segno di percentuale (%) ed un simbolo che definisce il tipo di parametro di output (ad esempio,%s). Se avete bisogno di emettere in output il segno di percentuale nel formato stringa, utilizzare la specifica di formato %%.

flags

Flag	Descrizione	Comportamento Predefinito
- (meno)	Giustificazione a sinistra all'interno della larghezza impostata	Giustificazione a destra
+ (più)	Output del segno + o - per i valori di tipi di segno	Il segno viene visualizzato solo se il valore è negativo
0 (zero)	Gli zeri vengono aggiunti prima di un valore di output all'interno del preset larghezza. Se **0** flag è specificato con un formato intero (**i**, **u**, **x**, **X**, **o**, **d**) e l'accuratezza della specifica è impostata (per esempio, %04.d), allora 0 viene ignorato.	Nulla viene aggiunto
spazio	Uno spazio viene mostrato prima di un valore di uscita, se è un segno e valore positivo	Gli spazi non sono inseriti
#	Se usato insieme con il formato **o**, **x** o **X**, allora prima del valore di output 0, viene aggiunto rispettivamente 0x o 0X .	Nulla viene aggiunto
	Se usato insieme con il formato **e**, **E**, **a** o **A**, il valore viene sempre visualizzato con un punto decimale.	Decimal point is shown only if there is a non-zero fractional part.
	Se usato insieme con il formato **g** o **G**, il flag definisce la presenza di un punto decimale nel valore di uscita ed impedisce il taglio degli zeri iniziali. Il flag # viene ignorato se utilizzato con i formati **c**, **d**, **i**, **u**, **s**.	Decimal point is shown only if there is a non-zero fractional part. Gli zeri iniziali sono tagliati fuori.

width

Un numero non negativo decimale che imposta il numero minimo di simboli di output del valore formattato. Se il numero di simboli di uscita è inferiore alla larghezza specificata, il corrispondente numero di spazi viene aggiunto da sinistra o destra a seconda dell'allineamento (flag -). Se c'è flag zero (0), il corrispondente numero di zeri viene aggiunto prima del valore di uscita. Se il numero di simboli di uscita è maggiore della larghezza specificata, il valore di uscita non viene mai tagliato.

Se un asterisco (*) viene specificato come larghezza, il valore di tipo int deve essere indicato nel punto corrispondente della lista dei parametri passati. Esso sarà utilizzato per specificare lunghezza del valore di uscita.

precision

Un numero non negativo decimale che imposta la precisione di uscita - numero di cifre dopo la virgola decimale. A differenza della specificazion della larghezza, una specifica di precisione può tagliare la parte frazionaria del tipo con o senza arrotondamento.

L'uso di specifica di precisione è diverso per diversi tipi di formato.

Tipi	Descrizione	Comportamento Predefinito
a, A	Specifiche di precisione impostano il numero di cifre dopo la virgola decimale.	Precisione di default - 6.
c, C	Non utilizzato	
d, i, u, o, x, X	Imposta il numero minimo di cifre di uscita. Se il numero di cifre in un parametro corrispondente è minore di questa precisione, vengono aggiunti zeri alla sinistra del valore di uscita. Il valore di uscita non viene tagliato, se il numero di cifre di uscita è maggiore della precisione specificata.	Precisione di default - 1.
e, E, f	Imposta il numero di cifre di uscita dopo un punto decimale. L'ultima cifra è arrotondata.	Precisione di default - 6. Se la precisione impostata è 0, o la parte parte decimale è assente, il punto decimale non viene mostrato.
g, G	Imposta il numero massimo di numeri significativi.	6 numeri significativi sono in output.
s	Imposta il numero di simboli di uscita di una stringa. Se la lunghezza della stringa supera la precisione, la stringa viene tagliata.	L'intera stringa viene emessa in output.

```
PrintFormat("1. %s", _Symbol);
PrintFormat("2. %.3s", _Symbol);
int length=4;
PrintFormat("3. %.*s", length, _Symbol);
/*
1. EURUSD
2. EUR
3. EURU
/
```

h | l | ll | I32 | I64

Specificazione dei formati di dati, passati come parametro.

Tipo di Parametro	Prefisso utilizzato	Identificatore Comune del Tipo
int	l (L minuscola)	d, i, o, x, or X
uint	l (L minuscola)	o, u, x, or X

Tipo di Parametro	Prefisso utilizzato	Identificatore Comune del Tipo
long	ll (due L minuscole)	d, i, o, x, or X
short	h	d, i, o, x, or X
ushort	h	o, u, x, or X
int	I32	d, i, o, x, or X
uint	I32	o, u, x, or X
long	I64	d, i, o, x, or X
ulong	I64	o, u, x, or X

type

Lo specificatore di tipo è l'unico campo obbligatorio per l'output formattato.

Symbol	Tipo	Formato di Output
c	int	Simbolo di tipo Short (Unicode)
C	int	Simbolo di tipo char (ANSI)
d	int	Segno decimale intero
i	int	Segno decimale intero
o	int	Ottale intero senza segno
u	int	Intero decimale senza segno
x	int	Esadecimale intero senza segno, con "abcdef"
X	int	Esadecimale intero senza segno, usando "ABCDEF"
e	double	Un valore reale nel formato [-] d.dddde[segno] ddd, dove d - una cifra decimale, dddd - una o più cifre decimali, ddd - un numero a tre cifre che determina la grandezza del segno dell'esponente, - più o meno
E	double	Simile al formato di e, tranne che il segno di esponente è in output dalla lettera maiuscola (**E** invece di e)
f	double	Un valore reale nel formato [-] dddd.dddd, dove dddd - una o più cifre decimali. Il numero di cifre prima del punto decimale dipende dalla grandezza del valore numerico. Il numero di cifre dopo la virgola decimale dipende dalla precisione richiesta.
g	double	Un valore di output reale in formato **f** o **e** dipende da quale output è più compatto.

Symbol	Tipo	Formato di Output
G	double	Un valore di output reale in formato **F** o **E** dipende da quale output è più compatto.
a	double	Un numero reale in formato [-] 0xh.hhhh **p**± dd, dove h.hhhh - mantissa in forma di cifre esadecimali, con "abcdef", dd - Uno o più cifre dell' esponente. Il numero di cifre decimali è determinato dalle specifiche di accuratezza
A	double	Un numero reale in formato [−]0xh.hhhh **P**±dd, dove h.hhhh - mantissa in forma di cifre esadecimali, usando "ABCDEF", dd - Uno o più cifre di esponente. Il numero di cifre decimali è determinato dalle specifiche di accuratezza
s	string	Stringa di output

Invece di PrintFormat() è possibile utilizzare printf().

Esempio:

```
void OnStart()
  {
//--- trade server name
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- account number
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- long value output
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: leverage = 1:%I64d",
               server,login,leverage);
//--- account currency
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- double value output with 2 digits after the decimal point
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: account equity = %.2f %s",
               server,login,equity,currency);
//--- double value output with mandatory output of the +/- sign
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: current result for open positions = %+.2f %s",
               server,login,profit,currency);
//--- double value output with variable number of digits after the decimal point
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: point value  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- int value output
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: current spread in points = %d ",
               _Symbol,spread);
//--- double value output in the scientific (floating point) format with 17 meaningful
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- double value output in the scientific (floating point) format with 17 meaningful
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- output using PrintFormat() with default accuracy
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- simple output using Print()
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* execution result
   MetaQuotes-Demo 1889998: leverage = 1:100
   MetaQuotes-Demo 1889998: account equity = 22139.86 USD
   MetaQuotes-Demo 1889998: current result for open positions = +174.00 USD
   EURUSD: point value  = 0.00001
   EURUSD: current spread in points = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

Vedi anche

StringFormat, DoubleToString, Tipi Reali (double, float)

ResetLastError

Consente di impostare il valore della variabile predefinita _LastError a zero.

```
void  ResetLastError();
```

Valore restituito

Nessun valore restituito.

Nota

Va notato che la funzione GetLastError() non azzera la variabile _LastError. Di solito la funzione ResetLastError() viene chiamata prima di chiamare una funzione, dopo la quale viene controllata l'apparizione dell' errore.

ResourceCreate

Crea una risorsa immagine basata su un insieme di dati. Ci sono due varianti della funzione:

Creazione di una risorsa basata su un file

```
bool  ResourceCreate(
   const string      resource_name,       // Nome della risorsa
   const string      path                 // Un percorso relativo, al file
   );
```

Creazione di una risorsa in base all'array di pixel

```
bool  ResourceCreate(
   const string      resource_name,       // Nome della risorsa
   const uint&       data[],              // Data impostata come un array
   uint              img_width,           // Lo spessore della risorsa immagine
   uint              img_height,          // L'altezza della risorsa immagine
   uint              data_xoffset,        // L'offset del verso destro orizzontale del
   uint              data_yoffset,        // L'offset del verso in basso verticale del
   uint              data_width,          // Lo spessore totale dell'immagine basato s
   ENUM_COLOR_FORMAT color_format         // Metodo di elaborazione del colore
   );
```

Parametri

resource_name

[in] Nome risorsa.

data[][]

[in] Un array uni-dimensionale o bi-dimensionale per creare un'immagine completa.

img_width

[in] La larghezza dell'area rettangolare dell'immagine in pixel che devono essere immessi nella risorsa sottoforma di un'immagine. Non può essere maggiore di *data_width* (valore).

img_height

[in] L'altezza dell'area rettangolare dell'immagine in pixel da collocare nella risorsa sottoforma di un'immagine.

data_xoffset

[in] L'offset orizzontale verso destra dell'area rettangolare dell'immagine.

data_yoffset

[in] L'offset verticale verso il basso dell'area rettangolare dell'immagine.

data_width

[in] Obbligatorio solo per array monodimensionali. Denota l'intera larghezza della immagine dal set di dati. Se *data_width*=0, è considerato essere uguale a *img_width*. Per array bi-dimensionali il parametro viene ignorato e viene assunto essere pari alla seconda dimensione dei *dati[]* dell'array.

color_format

[in] Metodo di elaborazione dei colori, da un valore dell'enumerazione ENUM_COLOR_FORMAT.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore chiamare la funzione GetLastError(). Possono verificarsi i seguenti errori:

- 4015 - ERR_RESOURCE_NAME_DUPLICATED (nomi identici della dinamica e delle risorse statiche)
- 4016 - ERR_RESOURCE_NOT_FOUND (la risorsa non è stata trovata)
- 4017 - ERR_RESOURCE_UNSUPPORTED_TYPE (questo tipo di risorsa non è supportato)
- 4018 - ERR_RESOURCE_NAME_IS_TOO_LONG (il nome della risorsa è troppo lungo)

Nota

Se la seconda versione della funzione viene chiamata per creare la stessa risorsa con diversa larghezza, altezza e parametri di slittamento, non crea una nuova risorsa, ma aggiorna semplicemente quella esistente.

La prima versione della funzione viene utilizzata per il caricamento delle immagini e suoni da file, e la seconda versione viene utilizzata solo per la creazione dinamica di immagini.

Le immagini devono essere in formato BMP con una profondità di colore di 24 o 32 bit. I suoni possono essere solo in formato WAV. La dimensione della risorsa non deve superare i 16 Mb.

ENUM_COLOR_FORMAT

Identificatore	Descrizione
COLOR_FORMAT_XRGB_NOALPHA	La componente del canale alfa viene ignorata
COLOR_FORMAT_ARGB_RAW	Componenti di colore non vengono gestiti dal terminale (devono essere correttamente impostati dall'utente)
COLOR_FORMAT_ARGB_NORMALIZE	Componenti colore vengono gestiti dal terminale

Vedi anche

Risorse, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

ResourceFree

La funzione elimina le risorse create dinamicamente (liberando la memoria allocata per essa).

```
bool  ResourceFree(
   const string  resource_name      // nome della risorsa
   );
```

Parametri

resource_name

[in] Resource il nome deve iniziare con "::".

Valore restituito

True se avviene con successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

ResourceFree() consente agli sviluppatori di applicazioni mql5 di gestire il consumo di memoria quando si lavora attivamente con le risorse. Oggetti grafici legati alla risorsa sono eliminati dalla memoria vengono visualizzati correttamente dopo la sua eliminazione. Tuttavia, oggetti grafici di nuova costruzione (OBJ_BITMAP e OBJ_BITMAP_LABEL) non saranno in grado di utilizzare la risorsa eliminata.

La funzione cancella solo le risorse dinamiche create dal programma.

Vedi anche

Risorse, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

ResourceReadImage

La funzione legge i dati dalla risorsa grafica creata dalla funzione ResourceCreate() o salvato in un file EX5 durante la compilazione.

```
bool  ResourceReadImage(
   const string      resource_name,       // nome risorse grafiche per la lettura
   uint&             data[],              // array di ricezione dati dalla risorsa
   uint&             width,               // per ricevere la lareghezza dell'immagine
   uint&             height,              // per ricevere l'altezza immagine nella ris
   );
```

Parametri

resource_name

[in] Nome della risorsa grafica contenente un'immagine. Per accedere alle proprie risorse, il nome viene usato in forma breve "::resourcename". Se scarichiamo una risorsa da un file EX5 compilato, il nome completo dovrebbe essere usato con il percorso relativo a directory mql5, nome file e risorse - "percorso\\nomefile.ex5::resourcename".

data[][]

[in] array mono o bi dimensionale per la ricezione di dati dalla risorsa grafica.

img_width

[out] Larghezza della risorsa grafica immagine, in pixel.

img_height

[out] Altezza della risorsa grafica immagine, in pixel.

Valore restituito

vero se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Se l'array *data[]* viene quindi utilizzato per la creazione di una risorsa grafica, _COLOR_FORMAT_ARGB_NORMALIZE or COLOR_FORMAT_XRGB_NOALPHAdevono essere usati i formati colore.

Se l'array *data[]* è bi-dimensionale e la sua seconda dimensione è minore della grandezza X(larghezza) della risorsa grafica, la funzione ResourceReadImage() restituice falso e la lettura non viene eseguita. Ma se la risorsa esiste, l'attuale grandezza dell'immagine viene restituita ai parametri di larghezza ed altezza. Ciò consentirà di effettuare un altro tentativo di ricezione dei dati dalla risorsa.

Vedi anche

Resource, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

ResourceSav

Consente di salvare una risorsa nel file specificato.

```
bool  ResourceSave(
   const string  resource_name      // Nome della risorsa
   const string  file_name          // Nome del File
   );
```

Parametri

resource_name

[in] Il nome della risorsa, deve iniziare con "::".

file_name

[in] Il nome del file relativo a MQL5\Files.

Valore restituito

true - in caso di successo, altrimenti false. Per le informazioni di errore chiamare GetLastError().

Nota

La funzione sovrascrive sempre un file e crea tutte le directory necessarie intermedie nel nome del file, se necessario.

Vedi anche

Risorse, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

SetReturnError

Imposta il codice che restituisce il processo del terminale quando si completa l'operazione.

```
void  SetReturnError(
   int ret_code      // codice di comletamento del terminale client
   );
```

Parametri

ret_code

[in] Il codice da restituire dal processo del terminale client al termine dell'operazione.

Valore di ritorno

Nessun valore di ritorno.

Nota

Impostazione dello specificato codice di ritorno *ret_code* usando la funzione SetReturnError() è utile per analizzare i motivi del completamento dell'operazione programmatica quando si lancia il terminale tramite la riga di comando.

Diversamente dalla funzione TerminalClose(), SetReturnError() non completa l'operazione del terminale. Invece, imposta solo il codice che restituisce il processo del terminale al suo completamento.

Se la funzione SetReturnError() viene chiamata più volte e/o da diversi programmi MQL5, il terminale restituisce l'ultimo codice di ritorno impostato.

Il codice impostato viene restituito al completamento del processo del terminale, ad eccezione dei seguenti casi:

- un errore critico si è verificato durante l'esecuzione;
- è stata chiamata la funzione TerminalClose(int ret_code) che ha emesso il comando di completamento dell'operazione terminale con un codice specificato.

Guarda anche

Esecuzione Programma, Errori di Runtime, Codici di Motivazione di Deinizializzazione, TerminalClose

SetUserError

Imposta la variabile predefinita _LastError nel valore pari a ERR_USER_ERROR_FIRST + user_error

```
void  SetUserError(
   ushort user_error,    // numero errore
   );
```

Parametri

user_error

[in] Error numero impostato da un utente.

Valore restituito

Nessun valore restituito.

Nota

Dopo che un errore è stato impostato utilizzando la funzione SetUserError(user_error), GetLastError() restituisce il valore pari a ERR_USER_ERROR_FIRST + user_error.

Esempio:

```
voidOnStart()
  {
//--- imposta il numero errore 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- ottiene l' ultimo codice di errore
   Print("GetLastError = ",GetLastError());
/*
   Result
   GetLastError = 65537
*/
  }
```

Sleep

La funzione sospende l'esecuzione dell'Expert Advisor o script all'interno di un intervallo specificato.

```
void  Sleep(
   int  millisecondi      // intervallo
   );
```

Parametri

millisecondi

[in] L'intervallo di ritardo in millisecondi.

Valore restituito

Nessun valore restituito.

Nota

La funzione Sleep() non può essere chiamata per indicatori personalizzati, in quanto gli indicatori vengono eseguiti nel thread di interfaccia e non devono rallentarlo. La funzione ha il controllo built-in di della flag dell'alt dell EA EA ogni 0,1 secondi.

TerminalClose

La funzione comanda il terminale a completare il funzionamento.

```
bool  TerminalClose(
   int ret_code      // codice di chiusura del terminale client
   );
```

Parametri

ret_code

[in] Codice di ritorno, restituito dal processo del terminale client al completamento dell'operazione.

Valore restituito

La funzione restituisce true in caso di successo, altrimenti - false.

Nota

La funzione TerminalClose() non ferma il terminale immediatamente, comanda solo al terminale di completare la sua operazione.

Il codice di un Expert Advisor che ha chiamato TerminalClose() deve avere tutte le disposizioni per il completamento immediato (ad esempio tutti i file precedentemente aperti devono essere chiusi in modo normale). La chiamata di questa funzione deve essere seguita dall'operatore return.

Il parametro *ret_code* consente di indicare il codice di ritorno necessario per l'analisi dei motivi della cessazione programma del funzionamento del terminale, quando parte dal prompt dei comandi.

Esempio:

```
//--- parametri di input
input int  tiks_before=500; // numero di ticks fino alla terminazione
input int  pips_to_go=15;   // distanza in pips
input int  seconds_st=50;   // numero di secondi dato all' Expert Advisor
//--- globali
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                                           |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," reason code = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                    |
//+------------------------------------------------------------------------------+
void OnTick()
  {
   static double first_bid=0.0;
```

```
   MqlTick       tick;
   double        distance;
//---
   SymbolInfoTick(_Symbol,tick);
   tick_counter++;
   if(first_bid==0.0)
     {
      launch_time=tick.time;
      first_bid=tick.bid;
      Print("first_bid =",first_bid);
      return;
     }
//--- distanza prezzo in pips
   distance=(tick.bid-first_bid)/_Point;
//--- mostra una notifica per monitorare l'operazione dell' EA
   string comm="Secondi passati dal momento d' inizio:\r\n\x25CF: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF ticks received: "+(string)tick_counter+" ;"+
               "\r\n\x25CF price went in points: "+StringFormat("%G",distance);
   Comment(comm);
//--- sezione per il controllo della condizione di chiusura del terminale
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // uscita dal contatore tick
   if(distance>pips_to_go)
      TerminalClose(1);    // va su per numero di pips uguale a pips_to_go
   if(distance<-pips_to_go)
      TerminalClose(-1);   // va giu per numero di pips uguale a pips_to_go
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // terminazione per timeout
//---
  }
```

Vedi anche

Funzionamento del Programma, Errori di esecuzione, Motivi per deinizializzazione

TesterStatistics

La funzione restituisce il valore del parametro statistico specificato calcolato sulla base dei risultati di test.

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // ID
   );
```

Parametri

statistic_id

[in] L' ID del parametro statistico dell'enumerazione ENUM_STATISTICS.

Valore restituito

Il valore del parametro statistico dai risultati di testing.

Nota

La funzione può essere richiamata all'interno di OnTester() o OnDeinit() nel tester. In altri casi il risultato è indefinito.

TesterStop

Fornisce il comando di completamento dell'operazione del programma quando si fa il testing.

```
void  TesterStop();
```

Valore di ritorno

Nessun valore di ritorno.

Nota

La funzione TesterStop() è progettata per uno spegnimento anticipato di routine di un EA su un agente di testing - per esempio, quando si raggiunge un numero specificato di trade perdenti o un livello di drawdown preimpostato.

La chiamata TesterStop() è considerata un normale completamento di un test, pertanto viene chiamata la funzione OnTester () e i valori delle intere statistiche di trading accumulate e criterio di ottimizzazione vengono sottoposti al tester di strategia.

La chiamata di ExpertRemove() nel tester di strategia significa anche normale completamento del test e consente di ottenere statistiche di trading, ma l'EA viene de-caricato dalla memoria dell'agente. In questo caso, eseguire un passaggio(un pass) sul prossimo set di parametri richiede tempo per ricaricare il programma. Pertanto, TesterStop() è un'opzione preferita per il completamento di routine di un test.

Guarda anche

Esecuzione Programma, Test delle Strategie di Trading, ExpertRemove, SetReturnError

TesterDeposit

La funzione speciale che emula i fondi depositati, durante un test. Può essere utilizzato in alcuni sistemi di gestione del denaro.

```
bool  TesterDeposit(
   double money      // somma depositata
   );
```

Parametri

soldi

[in] Soldi da depositare su un conto nella valuta del deposito.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false.

Guarda anche

TesterWithdrawal

TesterHideIndicators

Imposta la modalità di visualizzazione/occultamento degli indicatori utilizzati in un EA. La funzione è destinata alla gestione della visibilità degli indicatori utilizzati solo durante i test.

```
void  TesterHideIndicators(
   bool      hide     // flag
   );
```

Parametri

hide

[in] Flag per nascondere gli indicatori durante il test. Impostare true per nascondere gli indicatori creati, altrimenti false.

Valore di ritorno

Nessuno.

Nota

Per impostazione predefinita, tutti gli indicatori creati in un EA testato vengono visualizzati nel chart del testing visuale. Inoltre, questi indicatori vengono visualizzati sul chart che viene aperto automaticamente al termine del test. La funzione TesterHideIndicators() consente agli sviluppatori di implementare la possibilità di disabilitare la visualizzazione degli indicatori utilizzati.

Per disabilitare la visualizzazione di un indicatore applicato durante il test di un EA, chiamare TesterHideIndicators() uguale a true prima di creare l'handle di EA - tutti gli indicatori creati dopo ciò, sono contrassegnati con il flag "Nascondi" (hide). Questi indicatori non vengono visualizzati durante un test visuale e sul chart che viene automaticamente aperto al termine del test.

Per disabilitare la modalità Nascondi degli indicatori appena creati, chiama TesterHideIndicators() uguale a false. Solo gli indicatori generati direttamente dall'EA testato possono essere visualizzati sul chart dei test. Questa regola si applica solo ai casi in cui non è presente un singolo template in <cartella_dati>MQL5\Profiles\Templates.

Se la <cartella_dati>MQL5\Profiles\Templates contiene un modello(template) speciale <nome_EA>.tpl, solo gli indicatori di questo template vengono visualizzati durante un test visuale e sul chart di testing. In questo caso, non vengono visualizzati indicatori applicati nell'EA testato. Questo comportamento rimane anche se TesterHideIndicators() uguale a true viene chiamato nel codice EA.

Se la directory <cartella_dati>MQL5\Profiles\Templates non contiene particolari template <nome_EA>.tpl avendo invece tester.tpl, gli indicatori di tester.tpl e quelli dell' EA non disabilitati dalla funzione TesterHideIndicators() vengono visualizzati durante un testing visuale e sul chart di testing. Se non esiste un template tester.tpl, vengono invece utilizzati gli indicatori del template default.tpl.

Se il tester di strategia non trova un template adatto (<nome_EA>.tpl, tester.tpl o default.tpl), la visualizzazione degli indicatori applicati nell' EA è completamente gestita dalla funzione TesterHideIndicators().

Esempio:

```
bool CSampleExpert::InitIndicators(void)
```

```
  {
   TesterHideIndicators(true);
//--- crea indicatore MACD
   if(m_handle_macd==INVALID_HANDLE)
      if((m_handle_macd=iMACD(NULL,0,12,26,9,PRICE_CLOSE))==INVALID_HANDLE)
        {
         printf("Errore creazione indicatore MACD");
         return(false);
        }
   TesterHideIndicators(false);
//--- crea l'indicatore EMA e lo aggiunge alla collezione
   if(m_handle_ema==INVALID_HANDLE)
      if((m_handle_ema=iMA(NULL,0,InpMATrendPeriod,0,MODE_EMA,PRICE_CLOSE))==INVALID_H
        {
         printf("Errore creazione indicatore EMA");
         return(false);
        }
//--- con successo
   return(true);
  }
```

Guarda anche

IndicatorRelease

TesterWithdrawal

La funzione speciale per emulare il l'operazione di prelievo del denaro nel processo di testing. Può essere utilizzata in alcuni sistemi di gestione del risparmio.

```
bool  TesterWithdrawal(
   double money      // la somma da prelevare
   );
```

Parametri

denaro

[in] La somma di denaro che abbiamo bisogno di ritirare (nella valuta di deposito).

Valore restituito

In caso di successo, restituisce true, in caso contrario - false.

TranslateKey

Restituisce un carattere Unicode da un codice tasto virtuale considerando la lingua di input corrente e lo status dei tasti di controllo.

```
short  TranslateKey(
   int  key_code      // codice tasto per la ricezione del carattere Unicode
   );
```

Parametri

key_code

[in] Codice Tasto.

Valore di ritorno

Carattere Unicode in caso di successo della conversione. La funzione restituisce -1 in caso di errore.

Nota

La funzione usa ToUnicodeEx per convertire i tasti premuti da un utente in caratteri Unicode. Un errore può verificarsi nel caso in cui ToUnicodeEx non viene attivato - per esempio, quando si cerca di ricevere il carattere tasto SHIFT.

Esempio:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string& s
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- se il carattere introdotto viene convertito con successo in Unicode
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("Errore TranslateKey per key=",lparam);
     }
  }
```

Guarda anche

Eventi del Terminale Client, OnChartEvent

ZeroMemory

La funzione consente di ripristinare una variabile passata per riferimento.

```
void  ZeroMemory(
   void & variable      // variabile di reset
   );
```

Parametri

variabile

[In] [out] Variabile passata per riferimento che si vuole resettare (inizializzare con valori zero).

Valore restituito

Nessun valore restituito.

Nota

Se il parametro della funzione è una stringa, la chiamata sarà equivalente a NULL come valore.
Per i tipi semplici e loro array, nonché per strutture/classi composte di questi tipi, questo è un semplice reset.
Per gli oggetti che contengono le stringhe e gli array dinamici, viene chiamata ZeroMemory() per ogni elemento.
Per gli array non protetti dal modificatore const, questa è l'azzeramento di tutti gli elementi.
Per array di oggetti complessi, ZeroMemory() viene chiamata per ogni elemento.

ZeroMemory() non può essere applicata alle classi membri protetti o ereditati.

Gruppo di funzioni per lavorare con gli Array

Gli array possono essere al massimo di quattro dimensioni. Ogni dimensione è indicizzata da 0 a *dimensione_grandezza-1*. In un caso particolare di un array unidimensionale di 50 elementi, la chiamata del primo elemento verrà visualizzato come array [0], dell 'ultimo - come array [49].

Funzione	Azione
ArrayBsearch	Restituisce l' indice del primo elemento trovato nella prima dimensione dell' array
ArrayCopy	Copia un array in un altro
ArrayCompare	Restituisce il risultato del confronto tra due array di tipi semplici o strutture personalizzate senza oggetti complessi
ArrayFree	Libera il buffer di ogni array dinamico ed imposta a 0 il valore della dimensione zero.
ArrayGetAsSeries	Controlla la direzione di indicizzazione degli array
ArrayInitialize	Imposta tutti gli elementi di un array numerico in un singolo valore
ArrayFill	Riempie un array con il valore specificato
ArrayIsSeries	Controlla se un array è una serie temporale (_* timeseries)
ArrayIsDynamic	Controlla se un array è dinamico
ArrayMaximum	Cerca l' elemento con il valore massimo
ArrayMinimum	Cerca l' elemento con il valore minimo
ArrayPrint	Prints an array of a simple type or a simple structure into journal
ArrayRange	Restituisce il numero di elementi nella dimensione specifica dell'array
ArrayResize	Imposta la nuova grandezza nella prima dimensione dell' array
ArrayInsert	Inserisce il numero specificato di elementi da un array sorgente ad uno ricevente a partire da un indice specificato
ArrayRemove	Rimuove il numero specificato di elementi dall'array iniziando con un indice specificato
ArrayReverse	Inverte il numero specificato di elementi nell'array iniziando con un indice specificato
ArraySetAsSeries	Imposta la direzione di indicizzazione dell' array
ArraySize	Restituisce il numero di elementi dell'array
ArraySort	Ordinamento di array numerico per la prima dimensione
ArraySwap	Scambia il contenuto di due array dinamici dello stesso tipo

ArrayBsearch

Searches for a specified value in a multidimensional numeric array sorted ascending. Search is performed through the elements of the first dimension.

Per la ricerca in un array di tipo double

```
int  ArrayBsearch(
   const double&    array[],   // array per la ricerca
   double           value      // ciò che dev'essere cercato
   );
```

Per la ricerca in un array di tipo float

```
int  ArrayBsearch(
   const float&    array[],   // array per la ricerca
   float           value      // ciò che dev'essere cercato
   );
```

Per la ricerca in un array di tipo long

```
int  ArrayBsearch(
   const long&    array[],   // array per la ricerca
   long           value      // ciò che dev'essere cercato
   );
```

Per la ricerca in un array di tipo int

```
int  ArrayBsearch(
   const int&    array[],   // array per la ricerca
   int           value      // ciò che dev'essere cercato
   );
```

Per la ricerca in un array di tipo short

```
int  ArrayBsearch(
   const short&    array[],   // array per la ricerca
   short           value      // ciò che dev'essere cercato
   );
```

Per la ricerca in un array di tipo char

```
int  ArrayBsearch(
   const char&    array[],   // array per la ricerca
   char           value      // ciò che dev'essere cercato
   );
```

Parametri

array[]

[in] Array numerico per la ricerca.

valore

[in] Valore per la ricerca.

Valore restituito

La funzione restituisce l'indice di un elemento trovato. Se il valore desiderato non viene trovato, la funzione restituisce l'indice di un elemento più vicino in termini di valore.

Nota

La ricerca binaria elabora solo array ordinati. Per ordinare gli array numerici utilizzare la funzione ArraySort().

Esempio:

```
#property description "Lo Script è basato sui dati che l'indicatore RSI mostra"
#property description "quanto spesso il mercato era in"
#property description "area overbought ed oversold nell'intervallo di tempo specificat
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- parametri di input
input int                InpMAPeriod=14;                  // Periodo media mobile
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;     // Tipo di prezzo
input double             InpOversoldValue=30.0;           // Livello Oversold (_* Sc
input double             InpOverboughtValue=70.0;         // Livello Overbought (_*
input datetime           InpDateStart=D'2012.01.01 00:00'; // Data di inizio Analisi
input datetime           InpDateFinish=D'2013.01.01 00:00'; // Data di Fine Analisi
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   double rsi_buff[]; // array dei valori dell'indicatore
   int    size=0;     // grandezza dell' array
//--- riceve l'handle dell'indicatore RSI
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- fallimento nel ricevere l'handle dell'indicatore
      PrintFormat("Indicator handle receiving error. Codice Errore = %d",GetLastError
      return;
     }
//--- è in loop, finchè l'indicatore calcola tutti i suoi valori
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- esce se l'indicatore ha completato forzatamente l'operazione dello script
      if(IsStopped())
         return;
      //--- una pausa per permettere all'indicatore di calcolare tutti i suoi valori
      Sleep(10);
     }
```

```
//--- copia i valori dell'indicatore per un certo periodo di tempo
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori dell'indicatore. Codice Errore = %d
      return;
     }
//--- riceve la grandezza dell'array
   size=ArraySize(rsi_buff);
//--- ordina l'array
   ArraySort(rsi_buff);
//--- trova l'orario (in termini percetuali) in cui il mercato era in area Oversold
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- trova l'orario (in termini percetuali) in cui il mercato era in area Overbought
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//--- forma la stringa per visualizzare i dati
   string str="From "+TimeToString(InpDateStart,TIME_DATE)+" to "
              +TimeToString(InpDateFinish,TIME_DATE)+" il mercato era:";
   string str_ovb="in area overbought per il "+DoubleToString(ovb,2)+"% del tempo";
   string str_ovs="in area oversold per il "+DoubleToString(ovs,2)+"% del tempo";
//--- mostra i dati sul chart
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);
//--- redisegna il chart
   ChartRedraw(0);
//--- pausa
   Sleep(10000);
  }
//+------------------------------------------------------------------------------+
//| Mostra i commenti sull' angolo sinistro superiore del chart                  |
//+------------------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- crea l'etichetta
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- associa l'etichetta verso l'angolo in basso a sinistra
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- cambia la posizione del punto di ancoraggio
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- distanza dal punto di ancoraggio in direzione-X
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- distanza dal punto di ancoraggio in direzione-Y
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- testo dell'etichetta
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- colore del testo
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
```

```
//--- grandezza del testo
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

ArrayCopy

Copia un array in un altro.

```
int  ArrayCopy(
   void&        dst_array[],          // array di destinazione
   const void&  src_array[],          // array sorgente
   int          dst_start=0,          // indice d'inizio dal quale scrivere nell'array
   int          src_start=0,          // primo indica di un array sorgente
   int          count=WHOLE_ARRAY     // numero di elementi
   );
```

Parametri

dst_array[]

[out] Array di Destinazione

src_array[]

[in] Array di Origine

dst_start=0

[in] Indice di partenza dall'array di destinazione. Per impostazione predefinita, indice di partenza è 0.

src_start=0

[in] indice iniziale per l'array d' origine. Per impostazione predefinita, indice di partenza è 0.

count=WHOLE_ARRAY

[in] Numero di elementi che devono essere copiati. Per default, l'intero array viene copiato (count=WHOLE_ARRAY).

Valore restituito

Restituisce il numero di elementi copiati.

Nota

If count<0 or count>src_size-src_start, all the remaining array part is copied. Gli array vengono copiati da sinistra a destra. Per array series (_*serie), la posizione di partenza è correttamente definita, aggiustata per la copia da sinistra a destra.

Se gli array sono di tipi diversi, durante la copiatura essa cerca di trasformare ogni elemento di un array di origine nel tipo di array di destinazione. Un array di stringhe può essere copiato solo in un'array di stringhe. Array di classi e strutture contenenti oggetti che richiedono l'inizializzazione, non vengono copiati. Un array di strutture può essere copiato solo in un array dello stesso tipo.

For dynamic arrays with indexing as in timeseries, the size of a destination array is automatically increased to the amount of copied data (if the latter exceeds the array size). The destination array size is not decreased automatically.

Esempio:

```
#property description "L'indicatore evidenzia le candele che sono locali"
#property description "alti e bassi. Intervallo per la ricerca della lunghezza"
```

```
#property description "valori estremi devono essere trovati usando i parametri di inpu
//--- impostazioni indicatore
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot
#property indicator_label1  "Extremums"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- costanti predefinite
#define INDICATOR_EMPTY_VALUE 0.0
//--- parametri di input
input int InpNum=4; // Lunghezza semi-intervallo
//--- buffers indicatore
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- variabili globali
int    ExtStart=0; // indice della prima candela che non è un estremo
int    ExtCount=0; // numero di non-estremi nell'intervallo
//+------------------------------------------------------------------------------------+
//| Riempimento di candele non estreme                                                 |
//+------------------------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- riempimento di candele
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- specificare il valore che non è mostrato
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
```

```
//--- specificare i nomi dei buffer indicatore per la visualizzazione nella finestra d
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- imposta l'indicizzazione dritta nelle time series
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- Variabile di inizio calcolo barra
   int start=prev_calculated;
//--- il calcolo non viene effettuato per le prime InpNum*2 barre
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- se la barra è appena nata, controlla il prossimo potenziale estremo
   if(rates_total-start==1)
      start--;
//--- l'indice della barra che dev'essere controllato per l'estremo
   int ext;
//--- calcolo loop valore indicatore
   for(int i=start;i<rates_total-1;i++)
     {
      //--- inizialmente sulla barra i senza disegnarla
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- indice dell'estremo per il controllo
      ext=i-InpNum;
      //--- controllo per il massimo locale
```

```
      if(IsMax(high,ext))
        {
         //--- evidenzia di una candela estrema
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- evidenzia altre candele fino all'estremo con un colore neutro
         FillCandles(open,high,low,close);
         //--- cambia i colori della variabile
         ExtStart=ext+1;
         ExtCount=0;
         //--- passa alla prossima iterazione
         continue;
        }
      //--- controlla il minimo locale
      if(IsMin(low,ext))
        {
         //--- evidenzia di una candela estrema
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- evidenzia altre candele fino all'estremo con un colore neutro
         FillCandles(open,high,low,close);
         //--- cambia i valori della variabile
         ExtStart=ext+1;
         ExtCount=0;
         //--- passa alla prossima iterazione
         continue;
        }
      //--- incrementa il numero dei non-estremi all'intervallo
      ExtCount++;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Controlla se l'auttuale elemento dell'array è un alto locale                 |
//+------------------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- variabile inizio dell' intervallo
   int i=ind-InpNum;
//--- periodo di fine dell'intervallo
   int finish=ind+InpNum+1;
//--- controlla la prima metà dell'intervallo
```

```
   for(;i<ind;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- controlla la seconda metà dell'intervallo
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- questo èun estremo
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Controlla se i correnti elementi dell'array sono bassi locali                |
//+------------------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- variabile inizio dell' intervallo
   int i=ind-InpNum;
//--- variabile di fine intervallo
   int finish=ind+InpNum+1;
//--- controlla la prima metà dell'intervallo
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- controlla la seconda metà dell'intervallo
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- questo èun estremo
   return(true);
  }
```

ArrayCompare

La funzione restituisce il risultato del confronto di due array dello stesso tipo. Esso può essere utilizzato per confrontare gli array di tipi semplici o strutture personalizzate senza oggetti complessi, che sono le strutture personalizzate che non contengono stringhe, array dinamici, classi e le altre strutture con oggetti complessi.

```
int  ArrayCompare(
   const void&  array1[],           // primo array
   const void&  array2[],           // secondo array
   int          start1=0,           // offset iniziale nel primo array
   int          start2=0,           // offset iniziale nel secondo array
   int          count=WHOLE_ARRAY   // numero di elementi per il confronto
   );
```

Parametri

array1[]

[in] Primo array.

array2[]

[in] Secondo array.

start1=0

[in] L'indice iniziale dell'elemento nel primo array, da cui inizia il confronto. L'inice di inizio di default - 0.

start2=0

[in] L'indice iniziale dell'elemento nel secondo array, da cui inizia il confronto. L'inice di inizio di default - 0.

count=WHOLE_ARRAY

[in] Numero di elementi da confrontare. Tutti gli elementi di entrambi gli array partecipano al confronto per default (count =WHOLE_ARRAY).

Valore restituito

- -1, se array1 [] è meno di array2 []
- 0, se array1 [] è pari ad array2 []
- 1, se array1 [] è più di array2 []
- -2, Se si verifica un errore a causa di incompatibilità dei tipi di array comparati, o se start1, start2 o valori di conteggio portano all'uscita di fuori dall'array.

Nota

La funzione non restituirà 0 (gli array non saranno considerati uguali) se gli array si differenziano per dimensioni e count=WHOLE_ARRAY per il caso in cui un array è un fedele sottoinsieme di un altro. In questo caso, il risultato del confronto delle grandezze di quelli array sarà restituito: -1, se la grandezza dell'array1 [] è inferiore alla grandezza dell' array2 [] , Altrimenti 1.

ArrayFree

Libera il buffer di qualsiasi array dinamico ed imposta il valore della dimensione zero a 0.

```
void  ArrayFree(
   void&  array[]      // array
   );
```

Parametri

array[]

[in] Array Dinamico.

Valore restituito

Nessun valore restituito.

Nota

La necessità di utilizzare la funzione ArrayFree() potrebbe non presentarsi troppo spesso considerando che tutta la memoria utilizzata viene liberata tutt'in una volta ed il lavoro principale con gli array comprende l'accesso ai buffer indicatore. Le grandezze dei buffer sono gestite automaticamente dal sottosistema esecutivo del terminale.

In caso sia necessario gestire manualmente la memoria in un complesso ambiente dinamico dell'applicazione, la funzione ArrayFree() consente di liberare la memoria occupata dal già inutile array dinamico, esplicitamente ed immediatamente.

Esempio:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- costanti predefinite
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------------------+
//| Classe dialogo per lavorare con la memoria                                   |
//+------------------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- grandezza array
   int               m_arr_size;
   //--- arrays
   char              m_arr_char[];
   int               m_arr_int[];
   float             m_arr_float[];
   double            m_arr_double[];
```

```
   long              m_arr_long[];
   //--- etichette
   CLabel            m_lbl_memory_physical;
   CLabel            m_lbl_memory_total;
   CLabel            m_lbl_memory_available;
   CLabel            m_lbl_memory_used;
   CLabel            m_lbl_array_size;
   CLabel            m_lbl_array_type;
   CLabel            m_lbl_error;
   CLabel            m_lbl_change_type;
   CLabel            m_lbl_add_size;
   //--- bottoni
   CButton           m_button_add;
   CButton           m_button_free;
   //--- combo boxes
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- valore corrente del tipo di array dalla combo box
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- metodo di creazione della classe object
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- handler dell'evento chart
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- crea etichette
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- crea elementi di controllo
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- event handlers
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- metodi per lavorare con il corrente array
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------------------+
//| Memoria libera dell'array corrente                                           |
//+------------------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
  {
//--- resetta la grandezza dell'array
```

```
   m_arr_size=0;
//--- libera l'array
   if(m_combo_box_type_value==0)
      ArrayFree(m_arr_char);
   if(m_combo_box_type_value==1)
      ArrayFree(m_arr_int);
   if(m_combo_box_type_value==2)
      ArrayFree(m_arr_float);
   if(m_combo_box_type_value==3)
      ArrayFree(m_arr_double);
   if(m_combo_box_type_value==4)
      ArrayFree(m_arr_long);
  }
//+------------------------------------------------------------------------------+
//| Tenta di aggiungere memoria al corrente array                                |
//+------------------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- esce se la grandezza della memoria utilizzata supera la grandezza della memoria
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOR
      return(false);
//--- tentativo di allocare memoria secondo il tipo corrente
   if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
      return(false);
//--- memoria allocata
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Events handling                                                              |
//+------------------------------------------------------------------------------+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------------------+
//| Costruttore                                                                  |
//+------------------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
  {
  }
```

```
//+------------------------------------------------------------------------------+
//| Distruttore                                                                  |
//+------------------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------------------+
//| Metodo di creazione oggetto della classe                                     |
//+------------------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- crea l'oggetto della classe base
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- prepara le stringhe per le etichette
   string str_physical="Memory physical = "+(string)TerminalInfoInteger(TERMINAL_MEMOF
   string str_total="Memory total = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOTA
   string str_available="Memory available = "+(string)TerminalInfoInteger(TERMINAL_MEM
   string str_used="Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_USED)+
//--- crea le etichette
   if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_phy
      return(false);
   if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,12
      return(false);
   if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,str_
      return(false);
   if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,cl
      return(false);
   if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Array type =
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size =
      return(false);
   if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
      return(false);
   if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Chang
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to arra
      return(false);
//--- crea elementi di controllo
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlue)
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clrBl
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
      return(false);
//--- inizializza la variabile
```

```
   m_arr_size=0;
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Crea il bottone                                                              |
//+------------------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- crea il bottone
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- testo
   if(!button.Text(str))
      return(false);
//--- grandezza del font
   if(!button.FontSize(font_size))
      return(false);
//--- colore dell'etichetta
   if(!button.Color(clr))
      return(false);
//--- aggiunge il bottone agli elementi di controllo
   if(!Add(button))
      return(false);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Crea un box combo per la grandezza dell'array                                |
//+------------------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- crea il combo box
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- aggiunge elementi al combo box
   if(!m_combo_box_step.ItemAdd("100 000",100000))
      return(false);
   if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
      return(false);
//--- imposta gli elementi correnti del combo box
   if(!m_combo_box_step.SelectByValue(1000000))
      return(false);
```

```
//--- aggiunge il combo box agli elementi di controllo
   if(!Add(m_combo_box_step))
      return(false);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Crea un combo box per il tipo di array                                             |
//+------------------------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- crea il combo box
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- aggiunge elementi al combo box
   if(!m_combo_box_type.ItemAdd("char",0))
      return(false);
   if(!m_combo_box_type.ItemAdd("int",1))
      return(false);
   if(!m_combo_box_type.ItemAdd("float",2))
      return(false);
   if(!m_combo_box_type.ItemAdd("double",3))
      return(false);
   if(!m_combo_box_type.ItemAdd("long",4))
      return(false);
//--- imposta gli elementi correnti del combo box
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- memorizza i correnti elementi del combo box
   m_combo_box_type_value=3;
//--- aggiunge il combo box agli elementi di controllo
   if(!Add(m_combo_box_type))
      return(false);
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Crea un etichetta                                                                  |
//+------------------------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- crea un'etichetta
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- testo
   if(!lbl.Text(str))
      return(false);
```

```
//--- grandezza del font
   if(!lbl.FontSize(font_size))
      return(false);
//--- colore
   if(!lbl.Color(clr))
      return(false);
//--- aggiunge l'etichetta agli elementi di controllo
   if(!Add(lbl))
      return(false);
//--- avvenuto
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Handler dell'evento di click del bottone "Aggiungi"                                |
//+------------------------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- incrementa la grandezza dell'array
   m_arr_size+=(int)m_combo_box_step.Value();
//--- tentativo di allocare memoria per l'array corrente
   if(CurrentArrayAdd())
     {
      //--- memoria allocata, visualizza lo status corrente sullo schermo
      m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TE
      m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEM
      m_lbl_array_size.Text("Array size = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- fallimento nell'allocazione di memoria, visualizza il messaggio d'errore
      m_lbl_error.Text("Array is too large, error!");
      //--- restituisce la precedente grandezza dell'array
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------------------------+
//| Handler dell'evento di click del bottone "Libera"                                  |
//+------------------------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- Liberare la memoria dall'array corrente
   CurrentArrayFree();
//--- visualizza lo status corrente sullo schermo
   m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TERMI
   m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY
   m_lbl_array_size.Text("Array size = 0");
   m_lbl_error.Text("");
  }
```

```
//+------------------------------------------------------------------------------------+
//| Handler del cambio evento del combo box                                            |
//+------------------------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- verifica se il tipo dell' array è cambiato
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- libera la memoria dell'array corrente
      OnClickButtonFree();
      //--- lavora con un altro tipo di array
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- visualizza il nuovo tipo di array sullo schermo
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("Array type = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("Array type = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("Array type = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("Array type = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("Array type = long");
     }
  }
//--- Oggetto della classe CMemoryControl
CMemoryControl ExtDialog;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                          |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- crea il dialogo
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- lancia
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                                                 |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------------------------+
```

```
//| Funzione Expert evento chart                                                    |
//+---------------------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

ArrayGetAsSeries

Controlla la direzione dell' indice dell' array.

```
bool  ArrayGetAsSeries(
   const void&  array[]    // array per il controllo
   );
```

Parametri

array

[in] Array controllato.

Valore restituito

Restituisce vero, se l'array specificato, ha il flag AS_SERIES impostato, vale a dire l'accesso all' array viene eseguito dal posteriore all'anteriore come nelle timeseries. La timeseries differisce da un array usuale nel fatto che l'indicizzazione di elementi della timeseries viene eseguita dalla fine all'inizio (dai dati più recenti a quelli più vecchi).

Nota

Per verificare se un array appartiene alle timeseries, utilizzare la funzione ArrayIsSeries(). Arrai di dati sui prezzi passati come parametri di input nella funzione OnCalculate() non obbligatoriamente hanno la direzione di indicizzazione identica alle timeseries. La direzione di indicizzazione necessaria può essere impostata utilizzando la funzione ArraySetAsSeries().

Esempio:

```
#property description "L'indicatore calcola i valori assoluti della differenza tra"
#property description "i prezzi Open e Close o High e Low visualizzandoli in una sotto
#property description "come un istogramma."
//--- impostazioni indicatore
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- parametri di input
input bool InpAsSeries=true; // Direzione di indicizzazione nel buffere indicatore
input bool InpPrices=true;   // Calcolo dei prezzi (true - Open,Close; false - High,Lo
//--- buffer indicatore
double ExtBuffer[];
//+------------------------------------------------------------------------------+
//| Calcola i valori dell'indicatore                                             |
//+------------------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- variabile iniziale per il calcolo delle barre
```

```
   int start=prev_calculated;
//--- lavora all'ultima barra se i valori degli indicatori sono già stati calcolati al
   if(prev_calculated>0)
      start--;
//--- definisce la direzione di indicizzazione nell'array
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- sostituisce la direzione di indicizzazione con una diretta se necessario
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- calcola i valori dell'indicatore
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- lega i buffer indicatore
   SetIndexBuffer(0,ExtBuffer);
//--- imposta l'elemento di indicizzazione nel buffer indicatore
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- controlla per quali prezzi l'indicatore è calcolato
   if(InpPrices)
     {
      //--- Apre e Chiude i prezzi
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- imposta i colori dell'indicatore
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
      //--- Prezzi High e Low
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- imposta i colori dell'indicatore
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
```

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- calcola l'indicatore in base al valore della flag
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Accesso alle timeseries, ArraySetAsSeries

ArrayInitialize

La funzione inizializza un array numerico per un valore prestabilito.

For initialization of an array of char type

```
int  ArrayInitialize(
   char    array[],     // array inizializzato
   char    value        // valore che verrà impostato
   );
```

For initialization of an array of short type

```
int  ArrayInitialize(
   short   array[],     // array inizializzato
   short   value        // valore che verrà impostato
   );
```

For initialization of an array of int type

```
int  ArrayInitialize(
   int     array[],     // array inizializzato
   int     value        // valore che verrà impostato
   );
```

For initialization of an array of long type

```
int  ArrayInitialize(
   long    array[],     // array inizializzato
   long    value        // valore che verrà impostato
   );
```

For initialization of an array of float type

```
int  ArrayInitialize(
   float   array[],     // array inizializzato
   float   value        // valore che verrà impostato
   );
```

For initialization of an array of double type

```
int  ArrayInitialize(
   double  array[],     // array inizializzato
   double  value        // valore che verrà impostato
   );
```

For initialization of an array of bool type

```
int  ArrayInitialize(
   bool    array[],     // array inizializzato
   bool    value        // valore che verrà impostato
   );
```

For initialization of an array of uint type

```
int  ArrayInitialize(
   uint    array[],     // array inizializzato
   uint    value        // valore che verrà impostato
   );
```

Parametri

array[]

[out] Array numerico che deve essere inizializzato.

valore

[in] Nuovo valore che dovrebbe essere impostato per tutti gli elementi dell'array.

Valore restituito

Numero di elementi.

Nota

La funzione ArrayResize() permette di impostare la dimensione di un array con una riserva per un'ulteriore espansione senza la rilocazione fisica della memoria. Viene implementato per una migliore performance, perché le operazioni di delocalizzazione della memoria sono abbastanza lente.

L' inizializzazione dell' array con ArrayInitialize(array, init_val) non significa l'inizializzazione con lo stesso valore di elementi di riserva allocati per questo array. Ad ulteriore espansione dell' *array* utilizzando la funzione ArrayResize(), gli elementi verranno aggiunti alla fine dell'array, i loro valori saranno indefiniti ed in molti casi non saranno uguali ad *init_value*.

Esempio:

```
voidOnStart()
  {
//--- array dinamico
   double array[];
//--- imposta la grandezza dell'array per 100 elementi e riserviamo un buffer per alt
   ArrayResize(array,100,10);
//--- inizializza gli elementi dell'array con il valore EMPTY_VALUE=DBL_MAX
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Valori degli ultimi 10 elementi dopo l'inizializzazione");
   for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);
//--- espande l'array di 5 elementi
   ArrayResize(array,105);
   Print("Valori degli ultimi 10 elementi dopo ArrayResize(array,105)");
//--- i valori degli ultimi 5 elementi sono ottenuti dal buffer riserva
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

ArrayFill

La funzione riempie un array con il valore specificato.

```
void  ArrayFill(
   void&  array[],      // array
   int    start,         // indice d'inizio
   int    count,         // numero di elementi da riempire
   void   value          // valore
   );
```

Parametri

array[]

[out] Array di tipo semplice (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

start

[in] Indice iniziale. In tal caso, la specificata flag AS_SERIES, viene ignorata.

count

[in] Numero di elementi da riempire.

valore

[in] Valore con cui riempire l'array.

Valore restituito

Nessun valore restituito.

Nota

Quando la funzione ArrayFill() viene chiamata, la direzione indicizzazione normale (da sinistra a destra) è sempre implicita. Significa che la variazione dell'ordine di accesso agli elementi dell'array utilizzando la funzione ArraySetAsSeries() viene ignorata.

Un array multidimensionale è mostrato come uni-dimensionale quando viene elaborato dalla funzione ArrayFill(). Ad esempio, array [2][4] viene elaborato come array [8]. Di conseguenza, è possibile specificare l'indice dell'elemento iniziale, da essere uguale a 5 quando si lavora con questo array. Pertanto, la chiamata di ArrayFill(array, 5, 2, 3.14) per l'aray [2][4] riempie gli elementi array [1][1] ed array [1][2] con 3.14.

Esempio:

```
voidOnStart()
  {
//--- dichiara l'array dinamico
   int a[];
//--- imposta la grandezza
   ArrayResize(a,10);
//--- riempie i primi 5 elementi con 123
   ArrayFill(a,0,5,123);
//--- riempie i primi 5 elementi con 456
```

```
   ArrayFill(a,5,5,456);
//--- mostra valori
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

ArrayIsDynamic

La funzione controlla se un' array è dinamico.

```
bool  ArrayIsDynamic(
   const void&  array[]    // array controllato
   );
```

Parametri

array[]

[in] Array controllato.

Valore restituito

Restituisce vero se l'array selezionato è dinamico, altrimenti restituisce falso.

Esempio:

```
#property description "Questo indicatore non calcola i valori. Fa un singolo tentativo
#property description "applicare la chiamata alla funzione ArrayFree() a tre array: qu
#property description "un buffer indicatore. I risultati vengono mostrati nel journal
//--- impostazioni indicatore
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- variabili globali
double ExtDynamic[];   // array dinamico
double ExtStatic[100]; // array statico
bool   ExtFlag=true;   // flag
double ExtBuff[];      // buffer indicatore
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- alloca la memoria per l'array
   ArrayResize(ExtDynamic,100);
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizzato                             |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
```

```
  {
//--- esegue una singola analisi
   if(ExtFlag)
     {
      //--- tenta di liberare la memoria per gli array
      //--- 1. Array dinamico
      Print("+============================+");
      Print("1. Controlla l' array dinamico:");
      Print("Grandezza prima che la memoria venga liberata = ",ArraySize(ExtDynamic));
      Print("E' questo un array dinamico = ",ArrayIsDynamic(ExtDynamic) ? "Yes" : "No"
      //--- tentativo di liberare la memoria dell'array
      ArrayFree(ExtDynamic);
      Print("Grandezza dopo che la memoria è stata liberata = ",ArraySize(ExtDynamic))
      //--- 2. Array statico
      Print("2. Controlla l'array statico:");
      Print("Grandezza prima che la memoria venga liberata = ",ArraySize(ExtStatic));
      Print("E' questo un array dinamico = ",ArrayIsDynamic(ExtStatic) ? "Yes" : "No")
      //--- tentativo di liberare la memoria dell'array
      ArrayFree(ExtStatic);
      Print("Grandezza dopo che la memoria è stata liberata = ",ArraySize(ExtStatic));
      //--- 3. Buffer Indicatore
      Print("3. Controlla il buffer indicatore:");
      Print("Grandezza prima che la memoria venga liberata = ",ArraySize(ExtBuff));
      Print("E' questo un array dinamico = ",ArrayIsDynamic(ExtBuff) ? "Yes" : "No");
      //--- tentativo di liberare la memoria dell'array
      ArrayFree(ExtBuff);
      Print("Grandezza dopo che la memoria è stata liberata = ",ArraySize(ExtBuff));
      //--- cambia il valore della flag
      ExtFlag=false;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Accesso alle timeseries ed indicatori

ArrayIsSeries

La funzione controlla se un array è una timeseries.

```
bool  ArrayIsSeries(
   const void&  array[]    // array controllato
   );
```

Parametri

array[]

[in] Array controllato.

Valore restituito

Restituisce vero, se l'array controllato è un array di timeseries, altrimenti restituisce falso. Gli array passati come parametro alla funzione OnCalculate() devono essere verificati per l'ordine di accesso agli elementi dell'array da ArrayGetAsSeries().

Esempio:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers indicatore
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[] è una timeseries");
   else
      Print("open[] non è una timeseries!!!");
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Accesso alle timeseries ed indicatori

ArrayMaximum

Searches for the largest element in the first dimension of a multidimensional numeric array.

```
int  ArrayMaximum(
   const void&   array[],             // array per la ricerca
   int           start=0,             // indice dal quale iniziare il controllo
   int           count=WHOLE_ARRAY    // numero di elementi verificati
   );
```

Parametri

array[]

[in] l' array numerico, in cui viene effettuata la ricerca.

start=0

[in] Indice con cui avviare il controllo.

count=WHOLE_ARRAY

[in] Numero di elementi per la ricerca. Per impostazione predefinita, cerca in tutto l'array (count =WHOLE_ARRAY).

Valore restituito

La funzione restituisce l' indice di un elemento trovato tenendo conto dell' array seriale. Nel caso fallisse, restituisce -1.

Nota

The AS_SERIES flag value is taken into account while searching for a maximum.

Functions ArrayMaximum and ArrayMinimum accept any-dimensional arrays as a parameter. However, searching is always applied to the first (zero) dimension.

Esempio:

```
#property description "L'indicatore mostra candele di un timeframe più grande sul time
//--- impostazioni indicatore
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- costanti predefinite
#define INDICATOR_EMPTY_VALUE 0.0
//--- parametri di input
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Timeframe per il calcolo de
input datetime        InpDateStart=D'2013.01.01 00:00'; // Data di inizio analisi
//--- buffer indicatore per candele ribassiste
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- indicator buffers for bullish candlesticks
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variabili globali
datetime ExtTimeBuff[];      // più largo buffer di tempo del timeframe
int      ExtSize=0;          // grandezza del buffer tempo
int      ExtCount=0;         // indice dentro il buffer tempo
int      ExtStartPos=0;      // posizione iniziale per il calcolo dell'indicatore
bool     ExtStartFlag=true;  // flag ausiliaria per la ricezione della posizione inizi
datetime ExtCurrentTime[1];  // ultimo orario di creazione della barra più ampia
datetime ExtLastTime;        // ultimo orario dal timeframe più ampio, per cui viene e
bool     ExtBearFlag=true;   // flag per definire l'ordine di scrittura dei dato nei b
bool     ExtBullFlag=true;   // flag per definire l'ordine di scrittura dei dato nei b
int      ExtIndexMax=0;      // indice dell'elemento massimo nell' array
int      ExtIndexMin=0;      // indice dell'elemento minimo nell'array
int      ExtDirectionFlag=0; // direzione del movimento di prezzo per la corrente can
//--- spostamento tra il prezzo di apertura e di chiusura della candela per il corretto
```

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------------------+
//| Riempie la parte fondamentale della candela                                  |
//+------------------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- trova l'indice degli elementi massimi e minimi nell'array
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // massimo in High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // minimo in Low
//--- definisce quante barre dal corrente lasso di tempo devono essere compilate
   int count=fill_index-start+1;
//--- se il prezzo di chiusura alla prima barra eccede quella dell'ultima barra, la ca
   if(open[start]>close[last])
     {
      //--- se la candela è stata rialzista prima, cancella i valori del buffer indica
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- candela bearish(ribassista)
      ExtDirectionFlag=-1;
      //--- genera la candela
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- esce dalla funzione
      return;
     }
//--- se il prezzo di chiusura alla prima barra è inferiore a quello dell'ultima barra
   if(open[start]<close[last])
     {
      //--- se la candela è stata ribassista prima, cancella i valori del buffer indic
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- candela bullish(rialzista)
      ExtDirectionFlag=1;
      //--- genera la candela
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- esce dalla funzione
      return;
     }
//--- se siete in questa parte della funzione, il prezzo di apertura alla prima barra
//--- al prezzo di chiusura dell''ultima barra; tale candela è considerata ribassista
//--- se la candela è stata rialzista prima, cancella i valori del buffer indicatore
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- candela bearish
   ExtDirectionFlag=-1;
```

```
//--- se i prezzi close ed open sono uguali, utilizza lo shift per una corretta visual
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------------------+
//| Riempie la fine della candela                                    |
//+------------------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- non disegnare in caso di barra singola
   if(last-start==0)
      return;
//--- se il prezzo di chiusura alla prima barra eccede quella dell'ultima barra, la ca
   if(open[start]>close[last])
     {
      //--- genera la fine della candela
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- esce dalla funzione
      return;
     }
//--- se il prezzo di chiusura alla prima barra è inferiore a quello dell'ultima barra
   if(open[start]<close[last])
     {
      //--- genera la fine della candela
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- esce dalla funzione
      return;
     }
//--- se siete in questa parte della funzione, il prezzo di apertura alla prima barra
//--- al prezzo di chiusura dell''ultima barra; tale candela è considerata ribassista
//--- genera la fine della candela
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEr
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione di inizializzazione Indicatore Personalizzato  |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- imposta l'indicatore del periodo
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- mostra i dati dei prezzi in primo piano
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- lega i buffer indicatore
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- imposta alcuni valori proprietà, per creare l'indicatore
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // tipo di costruzione grafi
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // stile di disegno della li
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // spessore di disegno della
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

```
  {
//--- nel caso in cui non ci siano ancora barre calcolate
   if(prev_calculated==0)
     {
      //--- riceve l' orario di arrivo del frame's bar più grande
      if(!GetTimeData())
         return(0);
     }
//--- imposta l'indicizzazione diretta
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- variabile iniziale per il calcolo delle barre
   int start=prev_calculated;
//--- se la barra viene generata, ricalcola il valore dell'indicatore su di essa
   if(start!=0 && start==rates_total)
      start--;
//--- Il ciclo per calcolare i valori degli indicatori
   for(int i=start;i<rates_total;i++)
     {
      //--- riempie gli elementi del buffer indicatore per valori vuoti
      FillIndicatorBuffers(i);
      //--- esegue il calcolo per le barre iniziando dalla data InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- definisce la posizione, dal quale i valori devono essere mostrati, per
         if(ExtStartFlag)
           {
            //--- conserva il numero di barre iniziali
            ExtStartPos=i;
            //--- definisce la prima data dal timeframe più grande che eccedente time
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- modifica il valore del flag per non eseguire nuovamente questo bloc
            ExtStartFlag=false;
           }
         //--- controlla se ci sono ancora elementi nell'array
         if(ExtCount<ExtSize)
           {
            //--- attende il valore del timeframe corrente per raggiungere quello del
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- disegna la parte principale della candela (senza riempire l'area
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- riempie la fine della candela (l'area tra l'ultima e la penultima
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

```
                //--- slitta la posizione iniziale per disegnare la prossima candela
                ExtStartPos=i;
                //--- incrementa il contatore dell'array
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- resetra i valori dell'array
             ResetLastError();
             //--- riceve l'ultima data dal timeframe più grande
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Data copy error, code = ",GetLastError());
                return(0);
               }
             //--- se la nuova data è successiva, arresta la generazione di candele
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- sgombera l'area tra l' ultima e penultima barra nel principale bu
                ClearEndOfBodyMain(i-1);
                //--- compila l'area utilizzando buffer indicatori ausiliari
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- slitta la posizione iniziale per disegnare la prossima candela
                ExtStartPos=i;
                //--- resetta il flag della direzione del prezzo
                ExtDirectionFlag=0;
                //--- memorizza la nuova ultima data
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- genera la candela
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Controlla la correttezza del periodo dell'indicatore specificato             |
//+------------------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- il periodo dell'indicatore deve eccedere il timeframe con il quale esso viene v
```

```
   if(current_period>=high_period)
     {
      Print("Error! Il valore del periodo dell'indicatore deve superare il valore del
      return(false);
     }
//--- se il periodo dell'indicatore è di una settimana o mese, il periodo è corretto
   if(high_period>32768)
      return(true);
//--- converte i valori del periodo in minuti
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- il periodo dell'indicatore deve essere multiplo del timeframe su cui è visualizz
   if(high_period%current_period!=0)
     {
      Print("Error! Il valore del periodo dell' indicatore deve essere multiplo del va
      return(false);
     }
//--- il periodo dell'indicatore deve superare il timeframe su cui è visualizzato, di
   if(high_period/current_period<3)
     {
      Print("Error! Il periodo dell'indicatore dovrebbe superare l'attuale timeframe c
      return(false);
     }
//--- Il periodo indicatore è corretto per il timeframe corrente
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Riceve i dati temporali dal lasso di tempo più ampio |
//+------------------------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- copia tutti i dati per il tempo corrente
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- riceve il codice dell'errore
      int code=GetLastError();
      //--- stampa il messaggio d'errore
      PrintFormat("Errore copia dei dati! %s",code==4401
                  ? "Lo storico è ancora in fase di uploading!"
                  : "Code = "+IntegerToString(code));
      //--- restituisce false per fare un nuovo tentativo di download dei dati
      return(false);
     }
//--- riceve la grandezza dell'array
   ExtSize=ArraySize(ExtTimeBuff);
```

```
//--- imposta l'indice di loop per l'array a zero
   ExtCount=0;
//--- imposta la posizione della candela corrente sul timeframe, a zero
   ExtStartPos=0;
   ExtStartFlag=true;
//--- memorizza l'ultimo valore di tempo dal timeframe più ampio
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| La funzione costituisce la parte principale della candela. A seconda del flag |
//| del suo valore, la funzione definisce quali dati e array devono |
//| essere usati per la corretta visualizzazione.
//+------------------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- controlla il valore del flag
   if(flag)
     {
      //--- genera il corpo della candela
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- genera l'ombra della candela
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- genera il corpo della candela
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- genera l'ombra della candela
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------------+
//| La funzione costituisce l'estremità della candela. A seconda del valore della flag
//| la funzione definisce quali dati ed array devono |
//| essere usati per la corretta visualizzazione.
//+------------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- controlla il valore del flag
```

```
   if(flag)
     {
      //--- genera la fine del corpo della candela
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- genera la fine dell'ombra della candela
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- cambia il valore della flag in quello opposto
      flag=false;
     }
   else
     {
      //--- genera la fine del corpo della candela
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- genera la fine dell'ombra della candela
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- cambia il valore della flag in quello opposto
      flag=true;
     }
  }
//+------------------------------------------------------------------------------------
//| Cancella la fine della candela (l'area tra l'ultima e la penultima |
//| barra                                                              |
//+------------------------------------------------------------------------------------
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------------------------
//| Cancella la candela |
//+------------------------------------------------------------------------------------
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- controlla
   if(count!=0)
     {
      //--- riempie il buffer indicatore con valori vuoti
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------------------------
//| Genera la parte principale della candela                           |
//+------------------------------------------------------------------------------------
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

```
  {
//--- controlla
   if(count!=0)
     {
      //--- riempie il buffer indicatore con i valori
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------------------------------+
//| Genera la fine della candela                                                             |
//+------------------------------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- riempie il buffer indicatore con i valori
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------------------------------+
//| _* Riempie gli elementi del buffer indicatore per valori vuoti |
//+------------------------------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- imposta un valore vuoto nella cella del buffer indicatore
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

ArrayMinimum

Searches for the lowest element in the first dimension of a multidimensional numeric array.

```
int  ArrayMinimum(
   const void&   array[],             // array per la ricerca
   int           start=0,             // indice dal quale iniziare il controllo
   int           count=WHOLE_ARRAY    // numero di elementi controllati
   );
```

Parametri

array[]

[in] l' array numerico, in cui viene effettuata la ricerca.

start=0

[in] Indice con cui avviare il controllo.

count=WHOLE_ARRAY

[in] Numero di elementi per la ricerca. Per impostazione predefinita, cerca in tutto l'array (count =WHOLE_ARRAY).

Valore restituito

La funzione restituisce l' indice di un elemento trovato tenendo conto dell' array seriale. Nel caso fallisse, restituisce -1.

Nota

The AS_SERIES flag value is taken into account while searching for a minimum.

Functions ArrayMaximum and ArrayMinimum accept any-dimensional arrays as a parameter. However, searching is always applied to the first (zero) dimension.

Esempio:

```
#property description "L'indicatore mostra candele di un timeframe più grande sul time
//--- impostazioni indicatore
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- costanti predefinite
#define INDICATOR_EMPTY_VALUE 0.0
//--- parametri di input
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Timeframe per il calcolo de
input datetime        InpDateStart=D'2013.01.01 00:00'; // Data di inizio analisi
//--- buffer indicatore per candele ribassiste
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- indicator buffers for bullish candlesticks
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variabili globali
datetime ExtTimeBuff[];      // più largo buffer di tempo del timeframe
int      ExtSize=0;          // grandezza del buffer tempo
int      ExtCount=0;         // indice dentro il buffer tempo
int      ExtStartPos=0;      // posizione iniziale per il calcolo dell'indicatore
bool     ExtStartFlag=true;  // flag ausiliaria per la ricezione della posizione inizi
datetime ExtCurrentTime[1];  // ultimo orario di creazione della barra più ampia
datetime ExtLastTime;        // ultimo orario dal timeframe più ampio, per cui viene e
bool     ExtBearFlag=true;   // flag per definire l'ordine di scrittura dei dato nei b
bool     ExtBullFlag=true;   // flag per definire l'ordine di scrittura dei dato nei b
int      ExtIndexMax=0;      // indice dell'elemento massimo nell' array
int      ExtIndexMin=0;      // indice dell'elemento minimo nell'array
int      ExtDirectionFlag=0; // direzione del movimento di prezzo per la corrente can
//--- spostamento tra il prezzo di apertura e di chiusura della candela per il corretto
```

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------------------+
//| Riempie la parte fondamentale della candela                                  |
//+------------------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- trova l'indice degli elementi massimi e minimi nell'array
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // massimo in High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // minimo in Low
//--- definisce quante barre dal corrente lasso di tempo devono essere compilate
   int count=fill_index-start+1;
//--- se il prezzo di chiusura alla prima barra eccede quella dell'ultima barra, la ca
   if(open[start]>close[last])
     {
      //--- se la candela è stata rialzista prima, cancella i valori del buffer indica
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- candela bearish(ribassista)
      ExtDirectionFlag=-1;
      //--- genera la candela
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- esce dalla funzione
      return;
     }
//--- se il prezzo di chiusura alla prima barra è inferiore a quello dell'ultima barra
   if(open[start]<close[last])
     {
      //--- se la candela è stata ribassista prima, cancella i valori del buffer indic
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- candela bullish(rialzista)
      ExtDirectionFlag=1;
      //--- genera la candela
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- esce dalla funzione
      return;
     }
//--- se siete in questa parte della funzione, il prezzo di apertura alla prima barra
//--- al prezzo di chiusura dell''ultima barra; tale candela è considerata ribassista
//--- se la candela è stata rialzista prima, cancella i valori del buffer indicatore
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- candela bearish
   ExtDirectionFlag=-1;
```

```
//--- se i prezzi close ed open sono uguali, utilizza lo shift per una corretta visual
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------------------+
//| Riempie la fine della candela                                    |
//+------------------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- non disegnare in caso di barra singola
   if(last-start==0)
      return;
//--- se il prezzo di chiusura alla prima barra eccede quella dell'ultima barra, la ca
   if(open[start]>close[last])
     {
      //--- genera la fine della candela
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- esce dalla funzione
      return;
     }
//--- se il prezzo di chiusura alla prima barra è inferiore a quello dell'ultima barra
   if(open[start]<close[last])
     {
      //--- genera la fine della candela
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- esce dalla funzione
      return;
     }
//--- se siete in questa parte della funzione, il prezzo di apertura alla prima barra
//--- al prezzo di chiusura dell''ultima barra; tale candela è considerata ribassista
//--- genera la fine della candela
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEm
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione di inizializzazione Indicatore Personalizzato  |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- imposta l'indicatore del periodo
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- mostra i dati dei prezzi in primo piano
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- lega i buffer indicatore
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- imposta alcuni valori proprietà, per creare l'indicatore
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // tipo di costruzione grafi
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // stile di disegno della li
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // spessore di disegno della
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

```
  {
//--- nel caso in cui non ci siano ancora barre calcolate
   if(prev_calculated==0)
     {
      //--- riceve l' orario di arrivo del frame's bar più grande
      if(!GetTimeData())
         return(0);
     }
//--- imposta l'indicizzazione diretta
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- variabile iniziale per il calcolo delle barre
   int start=prev_calculated;
//--- se la barra viene generata, ricalcola il valore dell'indicatore su di essa
   if(start!=0 && start==rates_total)
      start--;
//--- Il ciclo per calcolare i valori degli indicatori
   for(int i=start;i<rates_total;i++)
     {
      //--- riempie gli elementi del buffer indicatore per valori vuoti
      FillIndicatorBuffers(i);
      //--- esegue il calcolo per le barre iniziando dalla data InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- definisce la posizione, dal quale i valori devono essere mostrati, per
         if(ExtStartFlag)
           {
            //--- conserva il numero di barre iniziali
            ExtStartPos=i;
            //--- definisce la prima data dal timeframe più grande che eccedente time
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- modifica il valore del flag per non eseguire nuovamente questo bloc
            ExtStartFlag=false;
           }
         //--- controlla se ci sono ancora elementi nell'array
         if(ExtCount<ExtSize)
           {
            //--- attende il valore del timeframe corrente per raggiungere quello del
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- disegna la parte principale della candela (senza riempire l'area
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- riempie la fine della candela (l'area tra l'ultima e la penultima
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

```
                //--- slitta la posizione iniziale per disegnare la prossima candela
                ExtStartPos=i;
                //--- incrementa il contatore dell'array
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- resetra i valori dell'array
             ResetLastError();
             //--- riceve l'ultima data dal timeframe più grande
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Data copy error, code = ",GetLastError());
                return(0);
               }
             //--- se la nuova data è successiva, arresta la generazione di candele
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- sgombera l'area tra l' ultima e penultima barra nel principale bu
                ClearEndOfBodyMain(i-1);
                //--- compila l'area utilizzando buffer indicatori ausiliari
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- slitta la posizione iniziale per disegnare la prossima candela
                ExtStartPos=i;
                //--- resetta il flag della direzione del prezzo
                ExtDirectionFlag=0;
                //--- memorizza la nuova ultima data
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- genera la candela
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Controlla la correttezza del periodo dell'indicatore specificato             |
//+------------------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- il periodo dell'indicatore deve eccedere il timeframe con il quale esso viene v
```

```
   if(current_period>=high_period)
     {
      Print("Error! Il valore del periodo dell'indicatore deve superare il valore del
      return(false);
     }
//--- se il periodo dell'indicatore è di una settimana o mese, il periodo è corretto
   if(high_period>32768)
      return(true);
//--- converte i valori del periodo in minuti
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- il periodo dell'indicatore deve essere multiplo del timeframe su cui è visualizz
   if(high_period%current_period!=0)
     {
      Print("Error! Il valore del periodo dell' indicatore deve essere multiplo del va
      return(false);
     }
//--- il periodo dell'indicatore deve superare il timeframe su cui è visualizzato, di
   if(high_period/current_period<3)
     {
      Print("Error! Il periodo dell'indicatore dovrebbe superare l'attuale timeframe c
      return(false);
     }
//--- Il periodo indicatore è corretto per il timeframe corrente
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Riceve i dati temporali dal lasso di tempo più ampio |
//+------------------------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- copia tutti i dati per il tempo corrente
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- riceve il codice dell'errore
      int code=GetLastError();
      //--- stampa il messaggio d'errore
      PrintFormat("Errore copia dei dati! %s",code==4401
                  ? "Lo storico è ancora in fase di uploading!"
                  : "Code = "+IntegerToString(code));
      //--- restituisce false per fare un nuovo tentativo di download dei dati
      return(false);
     }
//--- riceve la grandezza dell'array
   ExtSize=ArraySize(ExtTimeBuff);
```

```
//--- imposta l'indice di loop per l'array a zero
   ExtCount=0;
//--- imposta la posizione della candela corrente sul timeframe, a zero
   ExtStartPos=0;
   ExtStartFlag=true;
//--- memorizza l'ultimo valore di tempo dal timeframe più ampio
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| La funzione costituisce la parte principale della candela. A seconda del flag |
//| del suo valore, la funzione definisce quali dati e array devono |
//| essere usati per la corretta visualizzazione.
//+------------------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- controlla il valore del flag
   if(flag)
     {
      //--- genera il corpo della candela
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- genera l'ombra della candela
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- genera il corpo della candela
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- genera l'ombra della candela
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------------+
//| La funzione costituisce l'estremità della candela. A seconda del valore del flag,
//| la funzione definisce quali dati ed array devono |
//| essere usati per la corretta visualizzazione.
//+------------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- controlla il valore del flag
```

```
   if(flag)
     {
      //--- genera la fine del corpo della candela
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- genera la fine dell'ombra della candela
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- cambia il valore della flag in quello opposto
      flag=false;
     }
   else
     {
      //--- genera la fine del corpo della candela
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- genera la fine dell'ombra della candela
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- cambia il valore della flag in quello opposto
      flag=true;
     }
  }
//+------------------------------------------------------------------------------------
//| Cancella la fine della candela (l'area tra l'ultima e la penultima |
//| barra)
//+------------------------------------------------------------------------------------
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------------------+
//| Cancella la candela                                                          |
//+------------------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- controlla
   if(count!=0)
     {
      //--- riempie il buffer indicatore con valori vuoti
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------------------+
//| Genera la parte principale della candela |
//+------------------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

```
  {
//--- controlla
   if(count!=0)
     {
      //--- riempie il buffer indicatore con i valori
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------------------------+
//| Genera la fine della candela                                       |
//+------------------------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- riempie il buffer indicatore con i valori
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------------------------+
//| _* Riempie gli elementi del buffer indicatore per valori vuoti |
//+------------------------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- imposta un valore vuoto nella cella del buffer indicatore
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

ArrayPrint

Stampa un array di tipo semplice o una struttura semplice nel journal.

```
void  ArrayPrint(
   const void&   array[],             // array stampato
   uint          digits=_Digits,      // numero di posti decimali
   const string  separator=NULL,      // separatore dei valori dei campi della strutt
   ulong         start=0,             // indice del primo elemento stampato
   ulong         count=WHOLE_ARRAY,   // numero di elementi stampati
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_
   );
```

Parametri

array[]

[in] Array di tipo semplice o una struttura semplice.

digits=_Digits

[in] Il numero di posti decimali per il tipo real. Il valore di default è _Digits.

separator=NULL

[in] Separatore dei valori degli elementi del campo della struttura. Il valore di default NULL significa una linea vuota. Uno spazio è usato come separatore in questo caso.

start=0

[in] L'indice del primo elemento stampato dell'array. E' stampato dall'indice zero per default

count=WHOLE_ARRAY

[in] Numero degli elementi dell'arrat da essere stampati. L'intero array è visualizzato per default (count=WHOLE_ARRAY).

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] Combinazione delle impostazioni delle flags in output mode. Tutte le flags vengono abilitare per default.:

- ☐ ARRAYPRINT_HEADER - stama headers per l'array struttura
- ☐ ARRAYPRINT_INDEX - stampa l'indice al lato sinistro
- ☐ ARRAYPRINT_LIMIT - stampa solo i primi 100 e gli ultimi 100 elementi dell'array. Usa se vuoi stampare solo una parte di un array grande.
- ☐ ARRAYPRINT_ALIGN - abilita l'allineamento dei valori stampati - i numeri vengono allineati a destra, mentre le linee a sinistra.
- ☐ ARRAYPRINT_DATE - quando si stampa il datetime, stampa la data in formato dd.mm.yyyy (gg.mm.aaaa)
- ☐ ARRAYPRINT_MINUTES - quando si stampa il datetime, stampa l'orario in formato HH:MM
- ☐ ARRAYPRINT_SECONDS - quando si stampa il datetime, stampa l'orario in formato HH:MM:SS

Valore di ritorno

No

Nota

ArrayPrint() non stampa tutti i campi array della struttura nel journal - i campi di array e puntatore oggetto vengono saltati. Queste colonne semplicemente non vengono stampate per una presentazione più conveniente. Se avete bisogno di stampare tutti i campi della struttura, è necessario scrivere la propria funzione di stampa di massa con la formattazione desiderata.

Esempio:

```
// --- stampa i valori delle ultime 10 barre
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Check\n[time]\t[open]\t[high]\t[low]\t[close]\t[tick_volume]\t[spread]\t
      for(int i=0;i<10;i++)
        {
         PrintFormat("[%d]\t%s\t%G\t%G\t%G\t%G\t%G\t%G\t%I64d\t",i,
         TimeToString(rates[i].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
         rates[i].open,rates[i].high,rates[i].low,rates[i].close,
         rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
   else
      PrintFormat("CopyRates failed, error code=%d",GetLastError());
//--- esempio per la stampa
/*
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295         18110       10   17
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.11930 1.12747         17829        9   15
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744         13458       10    9
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194         15362        9   12
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172         16833        9   12
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052         15933        8   10
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528         11888        9    8
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915          7284       10    5
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904          8710        9    6
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263          8956        7    7
   Check
   [time] [open] [high] [low] [close] [tick_volume] [spread] [real_volume]
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295 18110 10 17300175000
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.1193 1.12747 17829 9 15632176000
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744 13458 10 9593492000
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194 15362 9 12352245000
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172 16833 9 12961333000
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052 15933 8 10720384000
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528 11888 9 8084811000
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915 7284 10 5087113000
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904 8710 9 6769629000
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263 8956 7 7192138000
*/
```

Guarda anche

FileSave, FileLoad

ArrayRange

La funzione restituisce il numero di elementi nella dimensione selezionata dell'array.

```
int  ArrayRange(
   const void&   array[],      // array per la verifica
   int           rank_index    // indice della dimensione
   );
```

Parametri

array[]

[in] Array controllato.

rank_index

[in] Indice della dimensione.

Valore restituito

Numero di elementi nella selezionata dell'array.

Nota

Poiché gli indici iniziano da zero, il numero delle dimensioni di un array è una maggiore dell'indice dell'ultima dimensione.

Esempio:

```
voidOnStart()
  {
//--- crea un array quadri-dimensionale
   double array[][5][2][4];
//--- imposta la grandezza della dimensione zero
   ArrayResize(array,10,10);
//--- stampa le dimensioni
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- riceve la grandezza della dimensione i
      temp=ArrayRange(array,i);
      //--- stampa
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- Risultato
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

ArrayResize

La funzione imposta una nuova grandezza per la prima dimensione

```
int  ArrayResize(
   void&  array[],              // array passato per riferimento
   int    new_size,             // nuova grandezza array
   int    reserve_size=0        // riserva valore grandezza (eccesso)
   );
```

Parametri

array[]

[out] Array per cui cambiare la grandezza.

new_size

[in] Nuova grandezza per la prima dimensione.

reserve_size=0

[in] Grandezza distribuita per avere la riserva.

Valore restituito

Se eseguita correttamente, restituisce il conteggio di tutti gli elementi contenuti nell' array dopo il ridimensionamento, in caso contrario, restituisce -1, e l'array non viene ridimensionato.

se ArrayResize() viene applicato ad un array, ad una timeseries o ad un buffer di indicatore, la dimensione dell'array rimane la stessa - questi array non verranno riallocati. In questo caso se *new_size* <=ArraySize (*array*), la funzione restituirà solo *new_size* ; altrimenti verrà restituito il valore di -1.

Nota

La funzione può essere applicata solo agli array dinamici. Va notato che non è possibile modificare la grandezza degli array dinamici assegnati come buffers di indicatore dalla funzione SetIndexBuffer(). Per i buffers indicatore, tutte le operazioni di ridimensionamento vengono eseguite dal sottosistema runtime del terminale.

Total amount of elements in the array cannot exceed 2147483647.

Con una frequente allocazione di memoria, si raccomanda di utilizzare un terzo parametro che imposta una riserva per ridurre il numero di allocazioni di memoria fisica. Tutte le chiamate successive di ArrayResize non portano ad una riallocazione della memoria fisica, ma solo modificano la grandezza della prima dimensione dell' array all'interno della memoria riservata. Va ricordato che il terzo parametro verrà utilizzato solo durante l'allocazione di memoria fisica. Ad esempio:

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

In questo caso, la memoria viene riallocata due volte, inizialmente, prima di inserire il ciclo di 3000-elementi (la grandezza dell'array sarà impostata a 1000), e la seconda volta poi con i pari a 2000. Se saltiamo il terzo parametro, ci saranno 2000 riallocazioni fisiche della memoria, che rallenteranno il programma.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- Contatori
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- Un array per la dimostrazione di una versione rapida
   double arr[];
   ArrayResize(arr,100000,100000);
//--- Controllare quanto velocemente funziona con la variante di riserva di memoria
   Print("--- Test Fast: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
//--- Impostare una nuova dimensione array che specifica la riserva di 100.000 element
      ArrayResize(arr,i,100000);
//--- Quando si raggiunge un numero tondo, mostrare la dimensione dell'array ed il ter
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-star
         start=now;
        }
     }
//--- Ora mostra, quanto è lenta la versione senza riserve di memoria
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Test Slow: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- Impostare una nuova dimensione array, ma senza la riserva aggiuntiva
      ArrayResize(slow,i);
//--- Quando si raggiunge un numero tondo, mostrare la dimensione dell'array ed il ter
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-st
         start=now;
        }
```

```
      }
   }
//--- Un semplice risultato dello script
/*
   Test_ArrayResize (EURUSD,H1)   --- Test Fast: ArrayResize(arr,100000,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(arr)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(arr)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(arr)=300000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   ---- Test Slow: ArrayResize(slow,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(slow)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(slow)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(slow)=300000 Time=228511 ms
*/
```

Vedi anche

ArrayInitialize

ArrayInsert

Inserisce il numero specificato di elementi da un array sorgente ad uno ricevente a partire da un indice specificato.

```
bool  ArrayInsert(
   void&        dst_array[],          // array ricevente
   const void&  src_array[],          // array sorgente
   uint         dst_start,            // indice dell'array ricevente da inserire
   uint         src_start=0,          // indice dell'array sorgente da copiare
   uint         count=WHOLE_ARRAY     // numero di elementi da inserire
   );
```

Parametri

dst_array[]

[in] [out] Array ricevente a cui devono essere aggiunti gli elementi.

src_array[]

[in] Array sorgente da cui devono essere aggiunti gli elementi.

dst_start

[in] Indice dell'array ricevente per l'inserimento di elementi dall'array sorgente.

src_start=0

[in] Indice dell'array sorgente, a partire dal quale gli elementi dell'array sorgente vengono presi per l'inserimento.

count

[in] Numero di elementi da aggiungere dall'array sorgente. WHOLE_ARRAY indica tutti gli elementi, dall'indice specificato fino alla fine dell'array.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError(). Possibili errori:

- 5052 - ERR_SMALL_ARRAY (i parametri *start* e/o *count* sono impostati in modo errato o l'array sorgente *src_array[]* è vuoto),
- 5056 - ERR_SERIES_ARRAY (l'array non può essere modificato, buffer indicatore),
- 4006 - ERR_INVALID_ARRAY (la copia su se stesso non è consentita, o gli array sono di tipo diverso, oppure esiste un array a grandezza fissa contenente oggetti classe o distruttori di strutture),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (l'array non contiene strutture POD e sta a significare che una semplice copia è impossibile),
- Si sono verificati errori quando la modifica della grandezza dell'array ricevente *dst_array[]* è stata fornita nella descrizione della funzioneArrayRemove() .

Nota

Se la funzione viene utilizzata per un array di grandezza fissa, la grandezza dell'array ricevente stesso *dst_array[]* non cambia. A partire dalla posizione *dst_start*, gli elementi dell'array ricevente

vengono spostati a destra (l'ultimo *counts* degli elementi "viene fuori"), mentre gli elementi copiati dall'array sorgente prendono il loro posto.

Non è possibile inserire gli elementi negli array dinamici designati come buffer indicatore dalla funzione SetIndexBuffer(). Per i buffer indicatori, tutte le operazioni di modifica della grandezza vengono eseguite dal sottosistema di esecuzione del terminale.

Nell'array sorgente, gli elementi vengono copiati a partire dall'indice *src_start*. La grandezza dell'array sorgente rimane invariata. Gli elementi da aggiungere all'array ricevente non sono collegamenti agli elementi dell'array sorgente. Ciò significa che le successive modifiche degli elementi in uno dei due array non si riflettono nel secondo.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma di script |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara l'array a dimensione fissa e riempie i valori
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- array sorgente
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- mostra gli array prima di inserire gli elementi
   Print("Prima di chiamare ArrayInsert()");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- inserisce 3 elementi dall'array sorgente e mostra il nuovo set dell'array riceve
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("Dopo aver chiamato ArrayInsert()");
   ArrayPrint(array_dest);
/*
  Risultato dell'esecuzione
   Prima di chiamare ArrayInsert()
   0 1 2 3 4 5 6 7 8 9
   Dopo aver chiamato ArrayInsert()
   0 1 2 3 10 11 12 7 8 9
*/
```

Guarda anche

ArrayRemove, ArrayCopy, ArrayResize, ArrayFree

ArrayRemove

Rimuove il numero specificato di elementi dall'array iniziando con un indice specificato.

```
bool  ArrayRemove(
   void&         array[],            // array di qualsiasi tipo
   uint          start,              // indice da cui parte la rimozione
   uint          count=WHOLE_ARRAY   // numero di elementi
   );
```

Parametri

array[]

[in][out] Array.

start

[in] Indice, a partire dal quale vengono rimossi gli elementi dell'array.

count=WHOLE_ARRAY

[in] Numero di elementi rimossi. Il valore WHOLE_ARRAY significa la rimozione di tutti gli elementi dall'indice specificato fino alla fine dell'array.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError(). Possibili errori:

- 5052 - ERR_SMALL_ARRAY (valore *start* troppo grande),
- 5056 - ERR_SERIES_ARRAY (l'array non può essere modificato, buffer indicatore),
- 4003 - ERR_INVALID_PARAMETER (valore *count* troppo grande),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (array a grandezza fissa contenente oggetti complessi con il distruttore),
- 4006 - ERR_INVALID_ARRAY (array a grandezza fissa contenente strutture o oggetti della classe con un distruttore).

Nota

Se la funzione viene utilizzata per un array di grandezza fissa, la grandezza dell'array non cambia: la "coda" rimanente viene fisicamente copiata nella posizione *start*. Per una comprensione accurata di come lavora la funzione, vedere l'esempio di seguito. Copia "fisica" significa che gli oggetti copiati non vengono creati chiamando il costruttore o l'operatore di copia. Invece, viene copiata la rappresentazione binaria dell'oggetto. Per questo motivo, non è possibile applicare la funzione ArrayRemove() all'array di grandezza fissa contenente oggetti con il distruttore (l'errore ERR_INVALID_ARRAY o ERR_STRUCT_WITHOBJECTS_ORCLASS viene attivato). Quando si rimuove un oggetto di questo tipo, il distruttore dovrebbe essere chiamato due volte - per l'oggetto originale e per la sua copia.

Non è possibile rimuovere elementi dagli array dinamici designati come buffer indicatore dalla funzione SetIndexBuffer(). Ciò provocherà l'errore ERR_SERIES_ARRAY. Per i buffer indicatori, tutte le operazioni di modifica della grandezza vengono eseguite dal sottosistema di esecuzione del terminale.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma di script |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara l'array a dimensione fissa e riempie i valori
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- mostra l'array prima di rimuovere gli elementi
   Print("Prima di chiamare ArrayRemove()");
   ArrayPrint(array);
// --- elimina 2 elementi dall'array e mostra il nuovo set
   ArrayRemove(array,4,2);
   Print("Dopo aver chiamato ArrayRemove()");
   ArrayPrint(array);
/*
  Risultato dell'esecuzione
  Prima di chiamare ArrayRemove()
   0 1 2 3 4 5 6 7 8 9
  Dopo aver chiamato ArrayRemove()
   0 1 2 3 6 7 8 9 8 9
*/
```

Guarda anche

ArrayInsert, ArrayCopy, ArrayResize, ArrayFree

ArrayReverse

Inverte il numero specificato di elementi nell'array iniziando con un indice specificato.

```
bool  ArrayReverse(
   void&         array[],            // array di qualsiasi tipo
   uint          start=0,            // indice da cui iniziare ad invertire l'array
   uint          count=WHOLE_ARRAY   // numero di elementi
   );
```

Parametri

array[]

[in][out] Array.

start=0

[in] Indice da cui inizia l'inversione dell'array.

count=WHOLE_ARRAY

[in] Numero di elementi invertiti. Se WHOLE_ARRAY, tutti gli elementi dell'array vengono spostati nel modo inverso a partire dall'indice specificato *start* fino alla fine dell'array.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false.

Nota

La funzione ArraySetAsSeries() non sposta fisicamente gli elementi dell'array. Invece, cambia solo la direzione di indicizzazione all'indietro, per disporre l'accesso agli elementi come nelle timeseries. La funzione ArrayReverse() sposta fisicamente gli elementi dell'array in modo che l'array sia "invertito".

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma di script |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara l'array a dimensione fissa e riempie i valori
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- mostra l'array prima di invertire gli elementi
   Print("Prima di chiamare ArrayReverse()");
   ArrayPrint(array);
//--- inverte 3 elementi nell'array e mostra il nuovo set
   ArrayReverse(array,4,3);
   Print("Dopo aver chiamato ArrayReverse()");
   ArrayPrint(array);
```

```
/*
  Risultato dell'esecuzione
  Prima di chiamare ArrayReverse()
   0 1 2 3 4 5 6 7 8 9
  Dopo aver chiamato ArrayReverse()
   0 1 2 3 6 5 4 7 8 9
*/
```

Guarda anche

ArrayInsert, ArrayRemove, ArrayCopy, ArrayResize, ArrayFree, ArrayGetAsSeries, ArraySetAsSeries

ArraySetAsSeries

La funzione imposta il flag AS_SERIES ad un oggetto di un array dinamico, E gli elementi verranno indicizzati come in TimeSeries.

```
bool  ArraySetAsSeries(
   const void&  array[],     // array per riferimento
   bool         flag         // true denota l'ordinamento invertito di indicizzazione
   );
```

Parametri

array[]

[in][out] Array numerico da impostare.

flag

[in] Direzione indicizzazione Array.

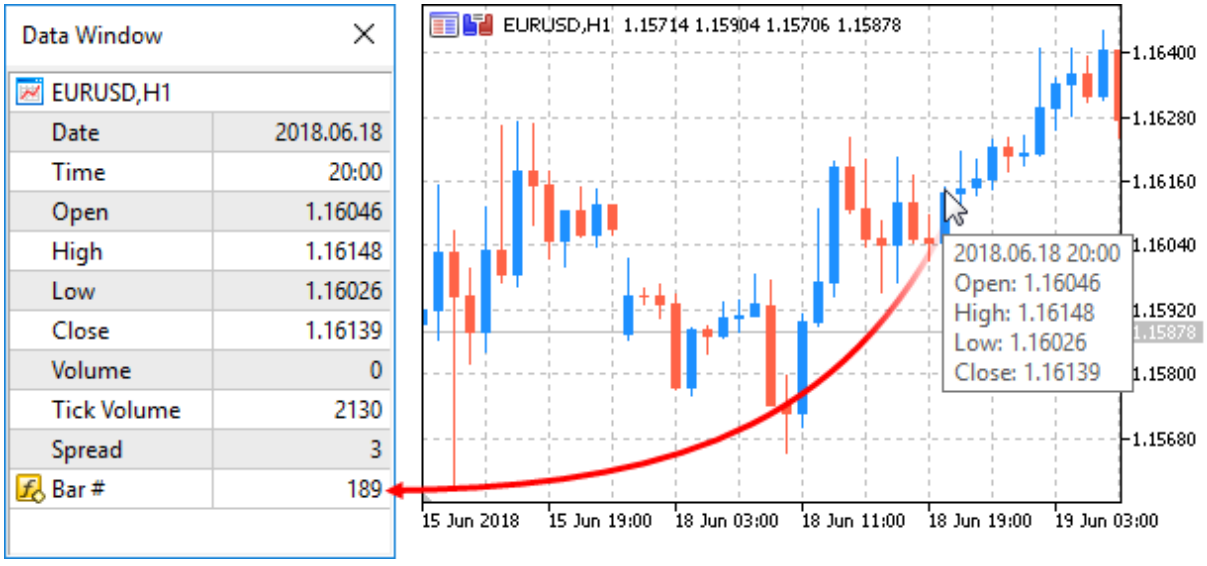
Valore restituito

La funzione restituisce true in caso di successo, altrimenti - false.

Nota

Il flag AS_SERIES non può essere impostato per array multidimensionali o array statici (array, le cui grandezze tra parentesi quadre sono preimpostate già in fase di compilazione). L' indicizzazione nelle timeseries si differenzia da array comune nel fatto che gli elementi delle timeseries sono indicizzati a partire dalla fine verso l'inizio (dai dati più recenti ai più vecchi).

Esempio: indicatore che mostra il numero di barre




```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot della Numerazione
```

```
#property indicator_label1  "Numerazione"
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- buffers indicatore
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- imposta indicizzazione per il buffer come nelle timeseries
   ArraySetAsSeries(NumerationBuffer,true);
//--- imposta l'accuratezza da mostrare in DataWindow
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- come il nome dell'array dell' indicatore viene visualizzato in DataWindow
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- memorizzamo l'ora di apertura della corrente barra
   static datetime currentBarTimeOpen=0;
//--- ripristiniamo l'accesso all' array time[] - facciamolo come nelle timeseries
   ArraySetAsSeries(time,true);
//--- Se il tempo della barra zero differisce da quello conservato
   if(currentBarTimeOpen!=time[0])
     {
     //--- enumeriamo tutte le barre da quella corrente alla profondità del grafico
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Accesso alle timeseries, ArrayGetAsSeries

ArraySize

La funzione restituisce il numero di elementi di un array selezionato.

```
int  ArraySize(
   const void&  array[]    // array verificato
   );
```

Parametri

array[]

[in] Array di qualsiasi tipo.

Valore restituito

Value of int type.

Nota

Per un array unidimensionale, il valore che deve essere restituito da ArraySize è uguale a quello di ArrayRange(array,0).

Esempio:

```
voidOnStart()
  {
//--- crea gli array
   double one_dim[];
   double four_dim[][10][5][2];
//--- grandezze
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- variabile ausiliaria
   int size;
//--- alloca la memoria senza backup
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. Array uni-dimensionale
   Print("+============================================================+");
   Print("Grandezze degli array:");
   Print("1. Array uni-dimensionale ");
   size=ArraySize(one_dim);
   PrintFormat("Grandezza della dimensione Zero = %d, Grandezza dell'array = %d",one_d
//--- 2. array multidimensionale
   Print("2. Array multidimensionale");
   size=ArraySize(four_dim);
   PrintFormat("Grandezza della dimensione Zero = %d, Grandezza dell'array = %d",four_
//--- grandezze delle dimensioni
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
```

```
   Print("Controllo:");
   Print("Dimensione Zero = Grandezza Array / (Prima dimensione * Seconda dimensione *
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. array uni-dimensionale con backup memoria
   Print("3. Array uni-dimensionale con un backup memoria");
//--- raddoppia il valore
   one_dim_size*=2;
//--- alloca la memoria con backup
   ArrayResize(one_dim,one_dim_size,reserve);
//--- stampa la grandezza
   size=ArraySize(one_dim);
   PrintFormat("Grandezza con backup = %d, Attuale grandezza dell'array = %d",one_dim_
  }
```

ArraySort

Sorts the values in the first dimension of a multidimensional numeric array in the ascending order.

```
bool  ArraySort(
   void&  array[]      // array per l'ordinamento
   );
```

Parametri

array[]

[in][out] Array numerido che deve essere messo in ordine.

Valore restituito

La funzione restituisce true in caso di successo, altrimenti - false.

Nota

An array is always sorted in the ascending order irrespective of the AS_SERIES flag value.

Functions ArraySort and ArrayBSearch accept any-dimensional arrays as a parameter. However, searching and sorting are always applied to the first (zero) dimension.

Esempio:

```
#property description "L'indicatore analizza i dati per l'ultimo mese e disegna tutte
#property description "e larghi volumi tick. L'array volume tick è stato ordinato"
#property description "per definire tali candele. Le candele hanno i volumi che compor
#property description "percentuale dell'array è considerata piccola. Le candle hanno i
#property description "l'ultima InpBigVolume percentuale dell'array sono considerate g
//--- impostazioni indicatore
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- costanti predefinite
#define INDICATOR_EMPTY_VALUE 0.0
//--- parametri di input
input int InpSmallVolume=15; // Valore percentuale di volumi piccoli (<50)
input int InpBigVolume=20;   // Valore percentuale di volumi grandi (>50)
//--- orario di inizio analisi (verrà slittato)
datetime ExtStartTime;
//--- buffers indicatore
double   ExtOpenBuff[];
double   ExtHighBuff[];
double   ExtLowBuff[];
```

```
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- valori limite di volume per la visualizzazione delle candele
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------------------+
//| Riceve i valori limite per i volumi tick                                     |
//+------------------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- variabili
   datetime stop_time;  // orario di fine della copia
   long     buff[];     // buffer per la copia
//--- l'orario di fine è quello corrente
   stop_time=TimeCurrent();
//--- l'orario d'inizio è un mese prima di quello corrente
   ExtStartTime=GetStartTime(stop_time);
//--- riceve i valori dei volumi tick
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- fallimento nel ricevere i dati, restituizioe false per lanciare il comando
      PrintFormat("Fallimento nel ricevere i valori dei volumi tick. Codice Errore = %
      return(false);
     }
//--- calcola la grandezza dell'array
   int size=ArraySize(buff);
//--- ordina l'array
   ArraySort(buff);
//--- definisce i valori dei bordi sinistro e destro per i volumi tick
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- esecuzione avvenuta
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Riceve i dati che sono di un mese in meno rispetto a quello passato          |
//+------------------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- converte l'orario finale in tipo variabile di struttura MqlDateTime
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- riceve i dati che sono un mese in meno
   if(temp.mon>1)
      temp.mon-=1;  // il mese corrente non è il primo dell'anno, quindi, il numero di
   else
     {
      temp.mon=12;  // il mese corrente è il primo dell'anno, quindi, il numero di que
```

```
      temp.year-=1; // mentre il numero dell'anno è uno in meno
     }
//--- il numero del giorno non eccederà 28
   if(temp.day>28)
      temp.day=28;
//--- restituisce la data ottenuta
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- controlla se i parametri di input soddisfano le condizioni
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("Parametri di input non corretti");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- imposta il valore che non verrà visualizzato
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- imposta le etichette per i buffer indicatore
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- controlla se le barre non gestite sono ancora presenti
   if(prev_calculated<rates_total)
     {
```

```
      //--- riceve nuovi valori dei bordi destro e sinistro per i volumi
      if(!GetVolumeBorders())
         return(0);
     }
//--- variabile di inizio per il calcolo della barra
   int start=prev_calculated;
//--- lavora all'ultima barra se i valori degli indicatori sono già stati calcolati al
   if(start>0)
      start--;
//--- imposta l'indicizzazione nelle timeseries
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- il loop di calcolo dei valori dell'indicatore
   for(int i=start;i<rates_total;i++)
     {
      //--- riempie le candele partendo dalla data iniziale
      if(ExtStartTime<=time[i])
        {
         //--- se il valore non è inferiore a quello del bordo destro, riempie la cand
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- riceve i dati per il disegno della candela
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- colore DodgerBlue
            ExtColorBuff[i]=0;
            //--- continua il loop
            continue;
           }
         //--- riempie la candela se il valore non supera quello del bordo sinistro
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- riceve i dati per il disegno della candela
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- Colore Orange
            ExtColorBuff[i]=1;
            //--- continua il loop
            continue;
           }
        }
```

```
      //--- imposta valori vuoti per le barre che non sono state incluse nel calcolo
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

ArrayBsearch

ArraySwap

Scambia il contenuto di due array dinamici dello stesso tipo. Per gli array multidimensionali, il numero di elementi in tutte le dimensioni tranne la prima, dovrebbe corrispondere.

```
bool  ArraySwap(
   void&  array1[],      // primo array
   void&  array2[]       // secondo array
   );
```

Parametri

array1[]

[in][out] Array di tipo numerico.

array2[]

[in][out] Array di tipo numerico.

Valore di ritorno

Restituisce true se ha successo, altrimenti false. In questo caso, GetLastError() restituisce il codice errore ERR_INVALID_ARRAY.

Nota

La funzione accetta array dinamici dello stesso tipo e delle stesse dimensioni tranne il primo. Per i tipi integer, il segno viene ignorato, vale a dire char== uchar)

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- array per conservare le quotazioni
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- ottiene i dati delle ultime 20 candele sul corrente timeframe
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("CopyRates(%s,0,0,20,rates) fallito, errore=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- imposta la grandezza dell'array per la quantità di dati copiati
   ArrayResize(source_array,copied);
//--- riempie l'array rate_array_1[] per i dati da rates[]
   for(int i=0;i<copied;i++)
     {
```

```
      source_array[i][0]=(double)rates[i].time;
      source_array[i][1]=rates[i].open;
      source_array[i][2]=rates[i].high;
      source_array[i][3]=rates[i].low;
      source_array[i][4]=rates[i].close;
      source_array[i][5]=(double)rates[i].tick_volume;
      source_array[i][6]=(double)rates[i].spread;
      source_array[i][7]=(double)rates[i].real_volume;
     }
//--- scambia dati tra source_array[] e dest_array[]
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("ArraySwap(source_array,rate_array_2) fallito, codice errore=%d",Get
      return;
     }
//--- assicurarsi che l'array di origine sia diventato zero dopo lo swap
   PrintFormat("ArraySwap() done: ArraySize(source_array)=%d",ArraySize(source_array))
//--- visualizzare i dati dell'array di destinazione dest_array[]
   ArrayPrint(dest_array);
  }
```

Guarda anche

ArrayCopy, ArrayFill, ArrayRange, ArrayIsDynamic

Matrici e vettori

Una matrice è un array bidimensionale di numeri di tipo double,float, o complex.

Un vettore è un array unidimensionale di numeri di tipo double, float o complex. Il vettore non ha alcuna indicazione se sia verticale o orizzontale. È determinato dal contesto d'uso. Ad esempio, l'operazione vettoriale Dot presuppone che il vettore sinistro sia orizzontale e quello destro verticale. Se è richiesta l'indicazione del tipo, è possibile utilizzare matrici a una riga o a una colonna. Tuttavia, questo non è generalmente necessario.

Matrici e vettori allocano dinamicamente la memoria per i dati. Infatti, matrici e vettori sono oggetti che hanno determinate proprietà, come il tipo di dati che contengono e le dimensioni. Le proprietà della matrice e del vettore possono essere ottenute utilizzando metodi come vector_a.Size(), matrix_b.Rows(), vector_c.Norm(), matrix_d.Cond() e altri. Qualsiasi dimensione può essere cambiata.

Quando si creano e inizializzano le matrici, vengono utilizzati i cosiddetti metodi statici (questi sono come i metodi statici di una classe). Per esempio: matrix::Eye(), matrix::Identity(), matrix::Ones(), vector::Ones(), matrix: :Zeros(), vector::Zeros(), matrix::Full(), vector::Full(), matrix::Tri().

Al momento, le operazioni matriciali e vettoriali non implicano l'uso del tipo di dati complex, poiché questa direzione di sviluppo non è ancora stata completata.

MQL5 supporta il passaggio di matrici e vettori a DLL. Questo permette l'importazione di funzioni che utilizzano i tipi corrispondenti, da variabili esterne.

Matrici e vettori sono passati a una DLL come puntatore a un buffer. Ad esempio, per passare una matrice di tipo float, il parametro corrispondente della funzione esportata dalla DLL deve avere un puntatore ad un buffer di tipo float.

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, cons
```

Oltre ai buffer, dovresti passare le dimensioni della matrice e del vettore per una corretta elaborazione.

Tutti i metodi matriciali e vettoriali sono elencati di seguito in ordine alfabetico.

Funzione	Azione	Categoria
Activation	Calcola i valori della funzione di attivazione e li scrive sul vettore/matrice passato	Apprendimento automatico
ArgMax	Restituisce l'indice del valore massimo	Statistica

Funzione	Azione	Categoria
ArgMin	Restituisce l'indice del valore minimo	Statistica
ArgSort	Restituisce l'indice ordinato	Manipolazioni
Assign	Copia una matrice, vettore o array con auto cast	Inizializzazione
Average	Calcola la media ponderata dei valori della matrice/vettore	Statistica
Cholesky	Calcola la decomposizione di Cholesky	Trasformazioni
Clip	Limita gli elementi di una matrice/vettore ad un determinato intervallo di valori validi	Manipolazioni
Col	Restituisce la colonna di un vettore. Scrive un vettore nella colonna specificata.	Manipolazioni
Cols	Restituisce il numero di colonne in una matrice	Caratteristiche
Compare	Confronta gli elementi di due matrici/vettori con la precisione specificata	Manipolazioni
CompareByDigits	Confronta gli elementi di due matrici/ vettori con la precisione di cifre importanti	Manipolazioni
Cond	Calcola il numero di condizione di una matrice	Caratteristiche
Convolve	Restituisce la discreta, convoluzione lineare di due vettori	Prodotti
Copy	Restituisce una copia della matrice/vettore dato	Manipolazioni
CopyIndicatorBuffer	Ottiene i dati del buffer dell'indicatore specificato nella quantità specificata in un vettore	Inizializzazione
CopyRates	Ottiene la serie storica della struttura MqlRates del periodo-simbolo specificato nella quantità specificata in una matrice o vettore	Inizializzazione

Funzione	Azione	Categoria
CopyTicks	Ottiene i tick da una struttura MqlTick in una matrice o un vettore	Inizializzazione
CopyTicksRange	Ottiene i tick da una struttura MqlTick in una matrice o un vettore entro l'intervallo di date specificato	Inizializzazione
CorrCoef	Calcola il coefficiente di correlazione di Pearson (coefficiente di correlazione lineare)	Prodotti
Correlate	Calcola la correlazione incrociata di due vettori	Prodotti
Cov	Calcola la matrice di covarianza	Prodotti
CumProd	Restituisce il prodotto cumulativo degli elementi matrice/vettore, compresi quelli lungo l'asse dato	Statistica
CumSum	Restituisce la somma cumulativa degli elementi matrice/vettore, compresi quelli lungo l'asse dato	Statistica
Derivative	Calcola i valori derivati della funzione di attivazione e li scrive nel vettore/matrice passato	Apprendimento automatico
Det	Calcola il determinante di una matrice invertibile quadrata	Caratteristiche
Diag	Estrae una diagonale o costruisce una diagonale della matrice	Manipolazioni
Dot	Prodotto scalare di due vettori	Prodotti
Eig	Calcola gli autovalori e gli autovettori destri di una matrice quadrata	Trasformazioni
EigVals	Calcola gli autovalori di una matrice generale	Trasformazioni
Eye	Restituisce una matrice con uno sulla diagonale e zero altrove	Inizializzazione

Funzione	Azione	Categoria
Fill	Riempire una matrice o un vettore esistente con il valore specificato	Inizializzazione
Flat	Accesso a un elemento matrice tramite un indice anziché due	Manipolazioni
Full	Crea e restituisce una nuova matrice riempita con il valore dato	Inizializzazione
GeMM	Il metodo GeMM (General Matrix Multiply) implementa la moltiplicazione generale di due matrici	Prodotti
HasNan	Restituisce il numero dei valori NaN in una matrice/vettore	Manipolazioni
Hsplit	Divide una matrice orizzontalmente in più sottomatrici. Stesso di Split con asse=0	Manipolazioni
Identity	Crea una matrice di identità della dimensione specificata	Inizializzazione
Init	Inizializzazione di matrice o vettore	Inizializzazione
Inner	Prodotto interno di due matrici	Prodotti
Inv	Calcola l'inverso moltiplicativo di una matrice invertibile quadrata con il metodo di Jordan-Gauss	Soluzioni
Kron	Restituisce il prodotto di Kronecker di due matrici, matrice e vettore, vettore e matrice o due vettori	Prodotti
LinearRegression	Calcola un vettore/matrice con i valori di regressione lineare calcolati	Statistica
Loss	Calcola i valori delle funzioni di perdita e li scrive nel vettore/matrice passato	Apprendimento automatico
LstSq	Restituisce la soluzione dei minimi quadrati delle equazioni algebriche lineari (per matrici non quadrate o degenerate)	Soluzioni

Funzione	Azione	Categoria
LU	Implementa una decomposizione LU di una matrice: il prodotto di una matrice triangolare inferiore e di una matrice triangolare superiore	Trasformazioni
LUP	Implementare una fattorizzazione LUP con permutazione parziale, che si riferisce alla decomposizione LU solo con permutazioni di riga: PA=LU	Trasformazioni
MatMul	Matrice con il prodotto di due matrici	Prodotti
Max	Restituisce il valore massimo in una matrice/vettore	Statistica
Mean	Calcola la media aritmetica dei valori degli elementi	Statistica
Median	Calcola la mediana degli elementi matrice/vettore	Statistica
Min	Restituisce il valore minimo in una matrice/vettore	Statistica
Norm	Restituisce la norma di una matrice o vettore	Caratteristiche
Ones	Crea e restituisce una nuova matrice riempita con uno	Inizializzazione
Outer	Calcola il prodotto esterno di due matrici o due vettori	Prodotti
Percentile	Restituisce il percentile specificato dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse specificato	Statistica
PInv	Calcola lo pseudo-inverso di una matrice con il metodo di Moore-Penrose	Soluzioni
Power	Eleva una matrice quadrata a una potenza intera	Prodotti
Prod	Restituisce il prodotto degli elementi matrice/vettore, che possono anche essere eseguiti per l'asse dato	Statistica

Funzione	Azione	Categoria
Ptp	Restituisce l'intervallo di valori di una matrice/vettore o dell'asse della matrice dato	Statistica
QR	Calcola la fattorizzazione qr di una matrice	Trasformazioni
Quantile	Restituisce il quantile specificato dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse specificato	Statistica
Rank	Restituisce il rango di una matrice utilizzando il metodo Gaussiano	Caratteristiche
RegressionMetric	Calcola la metrica dei regressori come errore di deviazione dalla linea di regressione costruita sull'array di dati specificato	Statistica
Reshape	Modifica la forma di una matrice senza modificare i suoi dati	Manipolazioni
Resize	Restituisce una nuova matrice con forma e dimensione modificata	Manipolazioni
Row	Restituisce la riga del vettore. Scrive il vettore nella riga specificata	Manipolazioni
Rows	Restituisce il numero di righe in una matrice	Caratteristiche
Size	Restituisce la dimensione del vettore	Caratteristiche
SLogDet	Calcola il segno e il logaritmo del determinante di una matrice	Caratteristiche
Solve	Risolve un'equazione matriciale lineare o un sistema di equazioni algebriche lineari	Soluzioni
Sort	Ordina per posizione	Manipolazioni
Spectrum	Calcola lo spettro di una matrice come l'insieme dei suoi autovalori dal prodotto AT*A	Caratteristiche
Split	Divide una matrice in più sottomatrici	Manipolazioni

Funzione	Azione	Categoria
Std	Restituisce la deviazione standard dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse specificato	Statistica
Sum	Restituisce la somma degli elementi della matrice/vettore, che possono anche essere eseguiti per l'asse dato (assi)	Statistica
SVD	Decomposizione ai valori singolari	Trasformazioni
SwapCols	Scambia le colonne in una matrice	Manipolazioni
SwapRows	Scambia le righe in una matrice	Manipolazioni
Trace	Restituisce la somma sulle diagonali della matrice	Caratteristiche
Transpose	Traspone (scambia gli assi) e restituisce la matrice modificata	Manipolazioni
Tri	Costruisce una matrice con uno sulla diagonale specificata e sotto di essa e zeri altrove	Inizializzazione
TriL	Restituisce una copia di una matrice con gli elementi sopra la diagonale k azzerati. Matrice triangolare inferiore	Manipolazioni
TriU	Restituisce una copia di una matrice con gli elementi sotto la diagonale k azzerati. Matrice triangolare superiore	Manipolazioni
Var	Calcola la varianza dei valori degli elementi della matrice/vettore	Statistica
Vsplit	Divide una matrice verticalmente in più sottomatrici. Stesso di Split con asse=1	Manipolazioni
Zeros	Crea e restituisce una nuova matrice piena di zeri	Inizializzazione

Tipi di matrice e vettore

Matrici e vettori sono tipi di dati speciali in MQL5 che consentono operazioni di algebra lineare. Esistono i seguenti tipi di dati:

- matrix — una matrice contenente elementi double.
- matrixf — una matrice contenente elementi float.
- matrixc — una matrice contenente elementi complex.
- vector — un vettore contenente elementi double.
- vectorf — un vettore contenente elementi float.
- vectorc — un vettore contenente elementi complex.

Le funzioni dei template supportano notazioni come matrix<double>, matrix<float>, vector<double>, vector<float> invece dei tipi corrispondenti.

Metodi di inizializzazione di matrici e vettori

Funzione	Azione
Eye	Restituisce una matrice con tutti uno sulla diagonale e zeri altrove
Identity	Crea una matrice identità della dimensione specificata
Ones	Crea e restituisce una nuova matrice riempita con uno
Zeros	Crea e restituisce una nuova matrice riempita con zeri
Full	Crea e restituisce una nuova matrice riempita con il valore dato
Tri	Costruisce una matrice con uno in corrispondenza e sotto la diagonale data e zeri altrove
Init	Inizializza una matrice o un vettore
Fill	Riempie una matrice o un vettore esistente con il valore specificato

Enumerazione per operazioni con matrici e vettori

Questa sezione descrive le enumerazioni che sono usate nei vari metodi matriciali e vettoriali.

ENUM_AVERAGE_MODE

Enumerazione dei tipi di media.

ID	Descrizione
AVERAGE_NONE	Nessuna media. I risultati sono forniti separatamente per ciascuna etichetta
AVERAGE_BINARY	Risultato etichetta 1 per la classificazione binaria
AVERAGE_MICRO	Risultato medio della matrice degli errori (matrice di confusione)
AVERAGE_MACRO	Risultato medio dei risultati delle matrici di errore di ciascuna etichetta
AVERAGE_WEIGHTED	Risultato medio ponderato

ENUM_VECTOR_NORM

Enumerazione di norme vettoriali per vector::Norm.

ID	Descrizione
VECTOR_NORM_INF	Norma infinito
VECTOR_NORM_MINUS_INF	Norma infinita negativa
VECTOR_NORM_P	Norma P

ENUM_MATRIX_NORM

Enumerazione delle norme matriciali per matrix::Norm e per ottenere la matrix::Cond numero di condizionamento della matrice.

ID	Descrizione
MATRIX_NORM_FROBENIUS	Norma di Frobenius
MATRIX_NORM_SPECTRAL	Norma spettrale
MATRIX_NORM_NUCLEAR	Norma nucleare
MATRIX_NORM_INF	Norma infinito

ID	Descrizione
MATRIX_NORM_P1	Norma P1
MATRIX_NORM_P2	Norma P2
MATRIX_NORM_MINUS_INF	Norma infinita negativa
MATRIX_NORM_MINUS_P1	Norma negativa P1
MATRIX_NORM_MINUS_P2	Norma negativa P2

ENUM_VECTOR_CONVOLVE

Enumerazione per convoluzioni vector::Convolve e correlazione incrociata vector::Correlate.

ID	Descrizione
VECTOR_CONVOLVE_FULL	Convoluzione completa
VECTOR_CONVOLVE_SAME	Convoluzione con lo stesso tipo
VECTOR_CONVOLVE_VALID	Convoluzione con tipo valido

ENUM_REGRESSION_METRIC

Enumerazione delle metriche di regressione per vector::RegressionMetric.

ID	Descrizione
REGRESSION_MAE	Errore Assoluto Medio
REGRESSION_MSE	Errore Quadratico Medio
REGRESSION_RMSE	Radice dell'Errore Quadratico Medio
REGRESSION_R2	R al Quadrato
REGRESSION_MAPE	Percentuale Errore Assoluto Medio
REGRESSION_MSPE	Percentuale Errore Quadratico Medio
REGRESSION_RMSLE	Radice dell'Errore Logaritmico Quadratico Medio
REGRESSION_SMAPE	Percentuale Errore Assoluto Medio Simmetrico
REGRESSION_MAXE	Errore assoluto Massimo
REGRESSION_MEDE	Errore Assoluto Mediano
REGRESSION_MPD	Devianza Media di Poisson
REGRESSION_MGD	Devianza Media Gamma

ID	Descrizione
REGRESSION_EXPV	Varianza Spiegata

ENUM_CLASSIFICATION_METRIC

Enumerazione delle metriche per i problemi di classificazione.

ID	Descrizione
CLASSIFICATION_ACCURACY	Qualità del modello in termini di accuratezza della previsione per tutte le classi
CLASSIFICATION_AVERAGE_PRECISION	Precisione media del modello
CLASSIFICATION_BALANCED_ACCURACY	Accuratezza della previsione bilanciata
CLASSIFICATION_F1	Punteggio F1. Media armonica tra precisione del modello e richiamo
CLASSIFICATION_JACCARD	Punteggio Jaccard
CLASSIFICATION_PRECISION	Accuratezza del modello nel predire i veri positivi per la classe target
CLASSIFICATION_RECALL	Completezza del modello
CLASSIFICATION_ROC_AUC	Area sotto la curva di errore
CLASSIFICATION_TOP_K_ACCURACY	Frequenza della corretta etichetta che compare nella parte superiore di k etichette previste

ENUM_LOSS_FUNCTION

Enumerazione per icalcoli della funzione di perdita vector::Loss.

ID	Descrizione
LOSS_MSE	Errore quadratico medio
LOSS_MAE	Errore medio assoluto
LOSS_CCE	Crossentropia Categorica
LOSS_BCE	Crossentropia Binaria
LOSS_MAPE	Errore in Percentuale Medio Assoluto
LOSS_MSLE	Errore logaritmico quadratico medio
LOSS_KLD	Divergenza di Kullback-Leibler
LOSS_COSINE	Somiglianza/prossimità del coseno

ID	Descrizione
LOSS_POISSON	Poisson
LOSS_HINGE	Hinge
LOSS_SQ_HINGE	Hinge al quadrato
LOSS_CAT_HINGE	Hinge Categorico
LOSS_LOG_COSH	Logaritmo del coseno iperbolico
LOSS_HUBER	Huber

ENUM_ACTIVATION_FUNCTION

Enumerazione per la funzione di attivazione vector::Activation e per il derivato della funzione di attivazione vector:::Derivative.

ID	Descrizione
AF_ELU	Unità Lineare Esponenziale
AF_EXP	Esponenziale
AF_GELU	Unità Lineare di Errore Gaussiana
AF_HARD_SIGMOID	Sigmoideo Hard
AF_LINEAR	Lineare
AF_LRELU	Unità Lineare Rettificata con Perdite
AF_RELU	Unità Lineare Rettificata
AF_SELU	Unità Lineare Esponenziale Scalata
AF_SIGMOID	Sigmoide
AF_SOFTMAX	Softmax funzione esponenziale normalizzata
AF_SOFTPLUS	Softplus
AF_SOFTSIGN	Softsign
AF_SWISH	Swish
AF_TANH	Funzione tangente iperbolica
AF_TRELU	Unità Lineare Rettificata con Soglia

ENUM_SORT_MODE

Enumerazione dei tipi di ordinamento per la funzione Sort.

ID	Descrizione
SORT_ASCENDING	Ordinamento ascendente
SORT_DESCENDING	Ordinamento discendente

ENUM_MATRIX_AXIS

Enumerazione per specificare l'asse in tutte le funzioni statistiche per le matrici.

ID	Descrizione
AXIS_NONE	L'asse non è specificato. Il calcolo viene eseguito su tutti gli elementi della matrice, come se fosse un vettore (vedere il metodo Flat).
AXIS_HORZ	Asse orizzontale
AXIS_VERT	Asse verticale

Initialization

Ci sono diversi modi per dichiarare e inizializzare matrici e vettori.

Funzione	Azione
Assign	Copia una matrice, vettore o array con auto cast
CopyIndicatorBuffer	Ottiene i dati del buffer dell'indicatore specificato nella quantità specificata in un vettore
CopyRates	Ottiene la serie storica della struttura MqlRates del periodo-simbolo specificato nella quantità specificata in una matrice o un vettore
CopyTicks	Ottiene i tick dalla struttura MqlTick in una matrice o vettore.
CopyTicksRange	Ottiene i tick dalla struttura MqlTick in una matrice o vettore all'interno dell'intervallo di date specificato.
Eye	Restituisce una matrice con tutti uno sulla diagonale e zeri altrove
Identity	Crea una matrice di identità della dimensione specificata
Ones	Crea e restituisce una nuova matrice riempita con tutti uno
Zeros	Crea e restituisce una nuova matrice riempita con zeri
Full	Crea e restituisce una nuova matrice riempita con un determinato valore
Tri	Costruisce una matrice con tutti uno sulla diagonale data e sotto di essa e zeri altrove
Init	Inizializza una matrice o un vettore
Fill	Riempie una matrice o un vettore esistente con il valore specificato

Dichiarazione senza specificare la dimensione (nessuna allocazione di memoria per i dati):

```
matrix          matrix_a;   // matrice di tipo double
matrix<double> matrix_a1;  // un altro modo per dichiarare una matrice double; può e
matrixf         matrix_a2;  // matrice float
matrix<float>  matrix_a3;  // matrice float
vector          vector_a;   // vettore double
vector<double> vector_a1;
vectorf         vector_a2;  // vettore float
vector<float>  vector_a3;
```

Dichiarazione con la dimensione specificata (con allocazione della memoria per i dati, ma senza inizializzazione):

```
matrix          matrix_a(128,128);           // i parametri possono essere sia costar
matrix<double> matrix_a1(InpRows,InpCols);  // o variabili
matrixf         matrix_a2(1,128);            // analogo ad un vettore orizzontale
```

```
  matrix<float>  matrix_a3(InpRows,1);        // analogo ad un vettore verticale
  vector         vector_a(256);
  vector<double> vector_a1(InpSize);
  vectorf        vector_a2(SomeFunc());       // la funzione SomeFunc restituisce un r
  vector<float>  vector_a3(InpSize+16);       // l'espressione può essere usata come p
```

Dichiarazione con inizializzazione (le dimensioni della matrice e del vettore sono determinate dalla sequenza di inizializzazione):

```
  matrix         matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix<double> matrix_a1=matrix_a;                    // devono essere matrici del
  matrixf        matrix_a2={{1,0,0},{0,1,0},{0,0,1}};
  matrix<float>  matrix_a3={{1,2},{3,4}};
  vector         vector_a={-5,-4,-3,-2,-1,0,1,2,3,4,5};
  vector<double> vector_a1={1,5,2.4,3.3};
  vectorf        vector_a2={0,1,2,3};
  vector<float>  vector_a3=vector_a2;                   // devono essere vettori del
```

Dichiarazione con inizializzazione:

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
...

  matrix  matrix_a(size_m,size_k,MatrixArange,-M_PI,0.1); //prima viene creata una mat
  matrixf matrix_a1(10,20,MatrixArange);                  // dopo aver creato la matri
  vector  vector_a(size,VectorArange,-10.0);              // dopo aver creato il vetto
  vectorf vector_a1(128,VectorArange);
```

Notare che le dimensioni della matrice o del vettore possono essere modificate, poiché la memoria per i dati è sempre dinamica.

Metodi statici

Metodi statici per la creazione di matrici e vettori della dimensione specificata, inizializzati in un certo modo:

```
matrix          matrix_a =matrix::Eye(4,5,1);
matrix<double> matrix_a1=matrix::Full(3,4,M_PI);
matrixf         matrix_a2=matrixf::Identity(5,5);
matrixf<float> matrix_a3=matrixf::Ones(5,5);
matrix          matrix_a4=matrix::Tri(4,5,-1);
vector          vector_a =vector::Ones(256);
vectorf         vector_a1=vector<float>::Zeros(16);
vector<float>   vector_a2=vectorf::Full(128,float_value);
```

Metodi per l'inizializzazione di matrici e vettori già creati:

```
matrix  matrix_a;
matrix_a.Init(size_m,size_k,MatrixArange,-M_PI,0.1);
matrixf matrix_a1(3,4);
matrix_a1.Init(10,20,MatrixArange);
vector  vector_a;
vector_a.Init(128,VectorArange);
vectorf vector_a1(10);
vector_a1.Init(vector_size,VectorArange,start_value,step);

matrix_a.Fill(double_value);
vector_a1.Fill(FLT_MIN);
matrix_a1.Identity();
```

Assign

Copia una matrice, vettore o array con auto cast.

```
bool matrix::Assign(
  const matrix<T>  &mat     // matrice copiata
   );
bool matrix::Assign(
  const void       &array[] // array copiato
   );
bool vector::Assign(
  const vector<T>  &vec     // vettore copiato
   );
bool vector::Assign(
  const void       &array[] // array copiato
   );
```

Parametri

m,v o array

[in] La matrice, il vettore o l'array da cui vengono copiati i valori.

Valore Restituito

Restituisce true in caso di successo, altrimenti — false.

Note

A differenza di Copy, il metodo Assign permette anche di copiare gli array. In questo caso, l'auto cast avviene, mentre la matrice o il vettore risultante si adatta alla dimensione dell'array copiato.

Esempio:

```
//--- copiare le matrici
  matrix a= {{2, 2}, {3, 3}, {4, 4}};
  matrix b=a+2;
  matrix c;
  Print("matrix a \n", a);
  Print("matrix b \n", b);
  c.Assign(b);
  Print("matrix c \n", a);

//---copia l'array nella matrice
  matrix double_matrix=matrix::Full(2,10,3.14);
  Print("double_matrix before Assign() \n", double_matrix);
  int int_arr[5][5]= {{1, 2}, {3, 4}, {5, 6}};
  Print("int_arr: ");
  ArrayPrint(int_arr);
  double_matrix.Assign(int_arr);
```

```
Print("double_matrix after Assign(int_arr) \n", double_matrix);
/*
 matrix a
 [[2,2]
  [3,3]
  [4,4]]
 matrix b
 [[4,4]
  [5,5]
  [6,6]]
 matrix c
 [[2,2]
  [3,3]
  [4,4]]

 double_matrix before Assign()
 [[3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]
  [3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]]

 int_arr:
     [,0][,1][,2][,3][,4]
 [0,]   1   2   0   0   0
 [1,]   3   4   0   0   0
 [2,]   5   6   0   0   0
 [3,]   0   0   0   0   0
 [4,]   0   0   0   0   0

 double_matrix after Assign(int_arr)
 [[1,2,0,0,0]
  [3,4,0,0,0]
  [5,6,0,0,0]
  [0,0,0,0,0]
  [0,0,0,0,0]]

*/
```

Vedi anche

Copy

CopyIndicatorBuffer

Ottiene i dati del buffer dell'indicatore specificato nella quantità specificata in un vettore.

I dati saranno copiati nel vettore posizionando l'elemento più vecchio all'inizio della memoria fisica allocata per il vettore. Ci sono tre opzioni di funzione.

Accesso tramite posizione iniziale e numero di elementi richiesti

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle indicatore
   ulong     buffer_index,         // numero del buffer dell'indicatore
   ulong     start_pos,            // posizione iniziale da copiare
   ulong     count                 // numero di elementi da copiare
   );
```

Accesso tramite data d'inizio e numero di elementi richiesti

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle indicatore
   ulong     buffer_index,         // numero del buffer dell'indicatore
   datetime  start_time,           // da quale data copiare
   ulong     count                 // numero di elementi da copiare
   );
```

Accesso tramite le date iniziale e finale dell'intervallo di tempo richiesto

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // handle indicatore
   ulong     buffer_index,         // numero del buffer dell'indicatore
   datetime  start_time,           // da quale data copiare
   datetime  stop_time             // fino a quale data copiare
   );
```

Parametri

indicator_handle

[in] L' handle dell'indicatore ottenuto dalla relativa funzione dell'indicatore.

buffer_index

[in] Il numero del buffer dell'indicatore.

start_pos

[in] Il numero dell'indice del primo elemento copiato.

count

[in] Il numero di elementi copiati.

start_time

[in] Ora della barra corrispondente al primo elemento.

stop_time

[in] Ora della barra corrispondente all'ultimo elemento.

Valore Restituito

La funzione restituisce 'true' in caso di successo o 'false' se si verifica un errore.

Note

Gli elementi dei dati copiati (il buffer dell'indicatore con indice buffer_index) sono contati dal presente al passato, e quindi la posizione di partenza uguale a 0 significa la barra corrente (il valore dell'indicatore per la barra corrente).

Quando si copia una quantità di dati sconosciuta, è necessario dichiarare un vettore senza specificare una dimensione (senza allocazione di memoria per i dati), poiché la funzione CopyBuffer() prova a ripartire la dimensione del vettore ricevente alla dimensione dei dati copiati.

Quando è necessaria la copia parziale dei valori dell'indicatore, è necessario utilizzare un vettore intermedio in cui viene copiata la quantità richiesta. Da questo vettore intermedio, è possibile copiare il numero richiesto di valori, membro per membro, ai posti richiesti del vettore ricevente.

Se si sta copiando una quantità predeterminata di dati, si consiglia di pre-dichiarare un vettore e specificare le sue dimensioni per evitare riallocazioni inutili della memoria.

Quando si richiedono dati da un indicatore, la funzione ritorna immediatamente false se le timeseries richieste non sono ancora state costruite o devono essere scaricate dal server, mentre il caricamento/ costruzione viene avviato.

Quando si richiedono dati da un'EA o da uno script, viene avviato il download dal server se il terminale non dispone dei dati appropriati localmente, o la costruzione delle timeseries necessarie inizia se i dati possono essere costruiti dalla storia locale, ma i timeframe richiesti non sono ancora pronti. La funzione restituisce la quantità che sarà pronta entro la scadenza del timeout.

Vedere anche

CopyBuffer

CopyRates

Ottiene la serie storica della struttura MqlRates del periodo-simbolo specificato nella quantità specificata in una matrice o un vettore. Gli elementi sono contati dal presente al passato, il che significa che la posizione di partenza uguale a 0 significa la barra corrente.

I dati vengono copiati in modo che l'elemento più vecchio sia posto all'inizio della matrice/vettore. Ci sono tre opzioni di funzione.

Accesso tramite posizione iniziale e numero di elementi richiesti

```
bool  CopyRates(
   string           symbol,       // nome del simbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinazione di flag per specificare la serie ric
   ulong            start,        // indice della barra iniziale da cui inizia la copi
   ulong            count         // quantità da copiare
   );
```

Accesso tramite data iniziale e numero di elementi richiesti

```
bool  CopyRates(
   string           symbol,       // nome del simbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinazione di flag per specificare la serie ric
   datetime         from,         // data d'inizio
   ulong            count         // quantità da copiare
   );
```

Accesso tramite data iniziale e finale dell'intervallo di tempo richiesto

```
bool  CopyRates(
   string           symbol,       // nome del simbolo
   ENUM_TIMEFRAMES  period,       // periodo
   ulong            rates_mask,   // combinazione di flag per specificare la serie ric
   datetime         from,         // data d'inizio
   datetime         to            // data di fine
   );
```

Parametri

symbol

[in] Simbolo.

period

[in] Periodo.

rates_mask

[in] L'enumerazione ENUM_COPY_RATES combinazione di flag specificando il tipo di serie richieste. Quando si copia un vettore, può essere specificato un solo valore dall'enumerazione ENUM_COPY_RATES, altrimenti si verifica un errore.

start

[in] Indice del primo elemento copiato.

count

[in] Numero degli elementi copiati.

from

[in] Ora della barra corrispondente al primo elemento.

to

[in] Ora della barra corrispondente all'ultimo elemento.

Valore Restituito

Restituisce true in caso di successo, altrimenti false in caso di errore.

Note

Se l'intervallo dei dati richiesti è completamente fuori dai dati disponibili sul server, la funzione restituisce false. Se i dati al di fuori TERMINAL_MAXBARS (numero massimo di barre sul grafico) vengono richiesti, la funzione restituisce ugualmente false.

Quando si richiedono dati da un'EA o da uno script, viene avviato il download dal server se il terminale non dispone dei dati appropriati localmente, o la costruzione delle Timeseries necessarie inizia se i dati possono essere costruiti dalla cronologia locale, ma non sono ancora pronti. La funzione restituisce la quantità che sarà pronta entro la scadenza del timeout, tuttavia il download della cronologia continua e la funzione restituisce più dati durante la successiva richiesta simile.

Quando si richiedono dati tramite data d'inizio e numero di elementi richiesti, vengono restituiti solo dati la cui data è inferiore a (prima) o uguale a quella specificata. L'intervallo è impostato e considerato fino a un secondo. In altre parole, la data di apertura di qualsiasi barra per cui viene restituito il valore (volume, spread, Apertura, Massimo, Minimo, Chiusura o Tempo) è sempre uguale o inferiore a quella specificata.

Quando si richiedono dati in un determinato intervallo di date, vengono restituiti solo i dati che rientrano nell'intervallo richiesto. L'intervallo è impostato e considerato fino a un secondo. In altre parole, il tempo di apertura di qualsiasi barra per cui viene restituito il valore (volume, spread, valore del buffer dell'indicatore, Apertura, Massimo, Minimo, Chiusura o Tempo) si trova sempre nell'intervallo richiesto.

Ad esempio, se il giorno corrente della settimana è Sabato, la funzione restituisce 0 quando si tenta di copiare i dati sull'intervallo di tempo settimanale impostando *start_time=Last_Tuesday* e *stop_time=Last_Friday* perché l'orario di apertura settimanale cade sempre di domenica, ma nessuna singola barra settimanale rientra nell'intervallo specificato.

Se è necessario ottenere il valore corrispondente della barra incompleta corrente, è possibile utilizzare il primo modulo di chiamata che indica *start_pos*=0 e *count*=1.

ENUM_COPY_RATES

L'enumerazione ENUM_COPY_RATES contiene i flag per specificare il tipo di dati da passare alla matrice o all'array. La combinazione di flag consente di ottenere diverse serie dalla cronologia in una richiesta. L'ordine delle righe nella matrice corrisponderà all'ordine dei valori nell'enumerazione

ENUM_COPY_RATES. In altre parole, la riga con i dati dei Massimi sarà sempre più alta nella matrice rispetto alla riga con i dati dei Minimi.

ID	Valore	Descrizione
COPY_RATES_OPEN	1	Serie dei prezzi d'Apertura
COPY_RATES_HIGH	2	Serie dei prezzi Massimi
COPY_RATES_LOW	4	Serie dei prezzi Minimi
COPY_RATES_CLOSE	8	Serie dei prezzi di Chiusura
COPY_RATES_TIME	16	Serie Temporali (ora di apertura della barra) Mettere il tempo nel vettore e nella matrice float (vectord e matrixf) causa perdite di ~100 secondi dato che la precisione di float la precisione è decisamente limitata e i numeri interi maggiori di 1<<24 non possono essere rappresentati esattamente in float.
COPY_RATES_VOLUME_TICK	32	Tick Volume
COPY_RATES_VOLUME_REAL	64	Volumi degli scambi
COPY_RATES_SPREAD	128	Spread
Combinazione		
COPY_RATES_OHLC	15	Serie Apertura, Massimo, Minimo e Chiusura
COPY_RATES_OHLCT	31	Serie Apertura, Massimo, Minimo, Chiusura e Ora

Esempio:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- ottenere le quotazioni nella matrice
  matrix matrix_rates;
  if(matrix_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLCT, 1, 10))
    Print("matrix rates: \n", matrix_rates);
  else
    Print("matrix_rates.CopyRates failed. Error ", GetLastError());
//--- controllo
  MqlRates mql_rates[];
  if(CopyRates(Symbol(), PERIOD_CURRENT, 1, 10, mql_rates)>0)
   {
    Print("mql_rates array:");
```

```
    ArrayPrint(mql_rates);
   }
  else
    Print("CopyRates(Symbol(), PERIOD_CURRENT,1, 10, mql_rates). Error ", GetLastError
//--- ottenere le quotazioni nel vettore = chiamata non valida
  vector vector_rates;
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLC, 1, 15))
    Print("vector_rates COPY_RATES_OHLC: \n", vector_rates);
  else
    Print("vector_rates.CopyRates COPY_RATES_OHLC failed. Error ", GetLastError());
/--- ottenere i prezzi di chiusura nel vettore
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_CLOSE, 1, 15))
    Print("vector_rates COPY_RATES_CLOSE: \n", vector_rates);
  else
    Print("vector_rates.CopyRates failed. Error ", GetLastError());
 };
/*
   matrix rates:
   [[0.99686,0.99638,0.99588,0.99441,0.99464,0.99594,0.99698,0.99758,0.99581,0.995280
    [0.99708,0.99643,0.99591,0.9955000000000001,0.99652,0.99795,0.99865,0.99764,0.996
    [0.9961100000000001,0.99491,0.99426,0.99441,0.99448,0.99494,0.9964499999999999,0.9
    [0.99641,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.995280000000000
    [1662436800,1662440400,1662444000,1662447600,1662451200,1662454800,1662458400,166
   mql_rates array:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2022.09.06 04:00:00 0.99686 0.99708 0.99611 0.99641          4463        0
   [1] 2022.09.06 05:00:00 0.99638 0.99643 0.99491 0.99588          4519        0
   [2] 2022.09.06 06:00:00 0.99588 0.99591 0.99426 0.99441          3060        0
   [3] 2022.09.06 07:00:00 0.99441 0.99550 0.99441 0.99464          3867        0
   [4] 2022.09.06 08:00:00 0.99464 0.99652 0.99448 0.99594          5280        0
   [5] 2022.09.06 09:00:00 0.99594 0.99795 0.99494 0.99697          7227        0
   [6] 2022.09.06 10:00:00 0.99698 0.99865 0.99645 0.99758         10130        0
   [7] 2022.09.06 11:00:00 0.99758 0.99764 0.99472 0.99581          7012        0
   [8] 2022.09.06 12:00:00 0.99581 0.99604 0.99360 0.99528          6166        0
   [9] 2022.09.06 13:00:00 0.99528 0.99570 0.99220 0.99259          6950        0
   vector_rates.CopyRates COPY_RATES_OHLC failed. Error 4003
   vector_rates COPY_RATES_CLOSE:
   [0.9931,0.99293,0.99417,0.99504,0.9968399999999999,0.99641,0.99588,0.99441,0.99464,
*/
```

Vedi anche

Accesso alle Timeseries e Indicatori, CopyRates

CopyTicks

Ottiene i tick dalla struttura MqlTick in una matrice o vettore. Gli elementi sono contati dal passato al presente, il che significa che il tick con l'indice 0 è il più vecchio. Per analizzare un tick, controllare il campo *flag* che mostra cosa esattamente è cambiato nel tick .

```
bool  matrix::CopyTicks(
   string           symbol,                 // nome del simbolo
   ulong            ticks_mask,             // maschera che indica il tipo di tick da r
   uint             flags=COPY_TICKS_ALL,   // flag che indica il tipo di tick da ricev
   ulong            from_msc=0,             // ora dalla quale i tick sono richiesti
   ulong            count=0                 // numero di tick da ricevere
   );
```

Metodo Vettoriale

```
bool  vector::CopyTicks(
   string           symbol,                 // nome del simbolo
   ulong            ticks_mask,             // maschera che indica il tipo di tick da r
   uint             flags=COPY_TICKS_ALL,   // flag che indica il tipo di tick da ricev
   ulong            from_msc=0,             // ora dalla quale i tick sono richiesti
   ulong            count=0                 // numero di tick da ricevere
   );
```

Parametri

symbol

[in] Simbolo.

ticks_mask

[in] Una combinazione di flag dall'enumerazione ENUM_COPY_TICKS indicando il contenuto dei dati richiesti. Quando si copia in un vettore, è possibile specificare un solo valore dall'enumerazione ENUM_COPY_TICKS, altrimenti si verificherà un errore.

flags

[in] Un flag che definisce il tipo di tick richiesto. COPY_TICKS_INFO indica i tick registrati dalle modifiche di Bid e/o Ask, COPY_TICKS_TRADE — tick con le modifiche di Last e Volume, COPY_TICKS_ALL — tutti i tick. Per qualsiasi tipo di richiesta, i valori del tick precedente vengono aggiunti ai restanti campi della struttura MqlTick.

from_msc

[in] Ora d'inizio dalla quale i tick sono richiesti. L'ora è specificata in millisecondi dal 01/01/1970. Se from_msc=0, l'ultimo numero di tick pari a 'count' vengono restituiti.

count

[in] Il numero dei tick richiesti. Se i parametri 'from_msc' e 'count' non sono specificati, tutte i tick disponibili, ma non più di 2000, saranno scritti.

Valore Restituito

Restituisce true in caso di successo, o false in caso di errore.

Note

La prima chiamata di CopyTicks() avvia la sincronizzazione del database tick del simbolo corrispondente memorizzato sul disco rigido. Se il database locale non fornisce tutti i tick richiesti, i tick mancanti verranno scaricati automaticamente dal server di trading. I tick specificati da ***from_msc*** in CopyTicks() fino al momento corrente saranno sincronizzati. Dopo di che, tutti i tiucke in arrivo per questo simbolo verranno aggiunti al database tick mantenendo così lo stato sincronizzato.

Se i parametri *from_msc* e *count* non sono specificati, tutti i tick disponibili, ma non più di 2000, saranno scritti nella matrice/ vettore.

Negli indicatori, il metodo CopyTicks() restituisce immediatamente il risultato: Quando viene chiamato da un indicatore, CopyTick() restituisce immediatamente tutti i tick disponibili di un simbolo e avvia la sincronizzazione del database dei tick se i dati disponibili non sono sufficienti. Tutti gli indicatori sullo stesso simbolo operano in un filo conduttore comune, così l'indicatore può non attendere il completamento della sincronizzazione. Dopo la sincronizzazione, CopyTicks() restituirà tutti i tick richiesti durante la chiamata successiva. Negli indicatori, la funzione OnCalculate() viene chiamata dopo l'arrivo di ogni tick.

Negli Expert Advisors e script, CopyTicks() può attendere il risultato per 45 secondi: diverso dagli indicatori, ogni Expert Advisor o script opera in un thread separato, e quindi può attendere fino a 45 secondi per il completamento della sincronizzazione. Se la quantità richiesta di tick non viene sincronizzata durante questo tempo, CopyTicks() restituirà i tick disponibili al timeout e continuerà la sincronizzazione. >OnTick() negli Expert Advisors non è un gestore di ogni tick, informa solo l'Expert Advisor sui cambiamenti nel mercato. Questo può essere una serie di modifiche: il terminale può ricevere contemporaneamente più tick, mentre OnTick() sarà chiamato solo una volta, per informare l'Expert Advisor circa l'ultimo stato del mercato.

Tasso di ritorno dei dati: il terminale memorizza gli ultimi 4096 tick per ogni strumento nella cache di accesso rapido (65536 tick per i simboli con la Profondità di Mercato in esecuzione). Le richieste riguardanti questi dati sono eseguite più velocemente. Se i ticks richiesti per la sessione di trading corrente sono oltre la cache, CopyTicks() chiama i ticks memorizzati nella memoria del terminale. Queste richieste richiedono più tempo per essere completate. Le richieste più lente sono quelle che richiedono i tick per altri giorni, poiché i dati vengono letti dall'unità in questo caso.

ENUM_COPY_TICKS

L'enumerazione ENUM_COPY_TICKS contiene i flag per specificare il tipo di dati da passare alla matrice o all'array. La combinazione di flag consente di ottenere diverse serie dalla cronologia in una richiesta. L'ordine delle righe nella matrice corrisponderà all'ordine dei valori nell'enumerazione ENUM_COPY_TICKS. In altre parole, la riga con i dati dei Massimi sarà sempre più alta nella matrice rispetto alla riga con i dati dei Minimi.

ID	Valore	Descrizione
COPY_TICKS_TIME_MS	1	Ora del tick in millisecondi
COPY_TICKS_BID	2	Prezzo del Bid
COPY_TICKS_ASK	4	Prezzo dell'Ask
COPY_TICKS_LAST	8	Ultimo prezzo (prezzo dell'ultimo contratto)

ID	Valore	Descrizione
COPY_TICKS_VOLUME	16	Volume dell'ultimo prezzo
COPY_TICKS_FLAGS	32	Flag dei tick

Analizzare i flag dei tick per scoprire quali dati sono cambiati:

- TICK_FLAG_BID — il tick ha cambiato il prezzo del bid
- TICK_FLAG_ASK — il tick ha cambiato il prezzo dell'ask
- TICK_FLAG_LAST — il tick ha cambiato il prezzo dell'ultima offerta
- TICK_FLAG_VOLUME — il tick ha cambiato il volume
- TICK_FLAG_BUY — il tick è il risultato di un contratto d'acquisto
- TICK_FLAG_SELL — il tick è il risultato di un contratto di vendita

Vedi anche

Accesso alle Timeseries e Indicatori, CopyTicks

CopyTicksRange

Ottiene i tick dalla struttura MqlTick in una matrice o vettore all'interno dell'intervallo di date specificato. Gli elementi sono contati dal passato al presente, il che significa che il tick con l'indice 0 è il più vecchio. Per analizzare un tick, controllare il campo *flag* che mostra cosa esattamente è cambiato nel tick .

```
bool  matrix::CopyTicksRange(
   string           symbol,                // nome del simbolo
   ulong            ticks_mask,            // maschera che indica il tipo di tick da i
   uint             flags=COPY_TICKS_ALL,  // flag che indica il tipo di tick da ricev
   ulong            from_msc=0,            // ora dalla quale i tick sono richiesti
   ulong            to_msc=0               // ora fino alla quale i tick sono richiest
   );
```

Metodo Vettoriale

```
bool  vector::CopyTicksRange(
   string           symbol,                // nome del simbolo
   ulong            ticks_mask,            // maschera che indica il tipo di tick da i
   uint             flags=COPY_TICKS_ALL,  // flag che indica il tipo di tick da ricev
   ulong            from_msc=0,            // ora dalla quale i tick sono richiesti
   ulong            to_msc=0               // ora fino alla quale i tick sono richiest
   );
```

Parametri

symbol

[in] Simbolo.

ticks_mask

[in] Una combinazione di flag dall'enumerazione ENUM_COPY_TICKS indicando il contenuto dei dati richiesti. Quando si copia in un vettore, è possibile specificare un solo valore dall'enumerazione ENUM_COPY_TICKS, altrimenti si verificherà un errore.

flags

[in] Un flag che definisce il tipo di tick richiesto. COPY_TICKS_INFO indica i tick registrati dalle modifiche di Bid e/o Ask, COPY_TICKS_TRADE — tick con le modifiche di Last e Volume, COPY_TICKS_ALL — tutti i tick. Per qualsiasi tipo di richiesta, i valori del tick precedente vengono aggiunti ai restanti campi della struttura MqlTick.

from_msc

[in] Ora d'inizio dalla quale i tick sono richiesti. L'ora è specificata in millisecondi dal 01/01/1970. Se il parametro 'from_msc' non è specificato, vengono restituiti i tick dall'inizio della cronologia. Saranno restituiti i tick con l'ora >= from_msc .

to_msc

[in] Ora fino alla quale i tick sono richiesti. L'ora è specificata in millisecondi dal 01/01/1970. Sono restituiti i tick con l'ora <= to_msc. Se il parametro to_msc non è specificato, vengono restituiti tutti i tick fino alla fine della cronologia.

Valore Restituito

Restituisce true in caso di successo o false in caso di errore. GetLastError() può restituire i seguenti errori:

- ERR_HISTORY_TIMEOUT — il timeout per la sincronizzazione tick è scaduto, la funzione ha restituito tutto quello che aveva.
- ERR_HISTORY_SMALL_BUFFER — il buffer statico è troppo piccolo. Solo la quantità che l'array può memorizzare è stata restituita.
- ERR_NOT_ENOUGH_MEMORY — memoria insufficiente per ricevere i dati storici dall'intervallo specificato in un array di tick dinamico. Impossibile allocare memoria sufficente per il tick array.

Analizzare i flag dei tick per scoprire quali dati sono cambiati:

- TICK_FLAG_BID — il tick ha cambiato il prezzo del bid
- TICK_FLAG_ASK — il tick ha cambiato il prezzo dell'ask
- TICK_FLAG_LAST — il tick ha cambiato il prezzo dell'ultima offerta
- TICK_FLAG_VOLUME — il tick ha cambiato il volume
- TICK_FLAG_BUY — il tick è il risultato di un contratto d'acquisto
- TICK_FLAG_SELL — il tick è il risultato di un contratto di vendita

Note

Il metodo CopyTicksRange() viene utilizzato per richiedere i tick esattamente dall'intervallo specificato. Ad esempio, i tick per un giorno specifico nella cronologia. CopyTicks() permette di specificare solo la data di inizio, per esempio, per ricevere tutti i tick dall'inizio del mese fino ad ora.

Vedi anche

Accesso alle Timeseries e Indicatori, CopyTicksRange

Eye

Una funzione statica. Costruisce una matrice avente una dimensione specificata con uno sulla diagonale principale e zeri altrove. Restituisce una matrice con uno sulla diagonale e zeri altrove.

```
static matrix matrix::Eye(
  const ulong  rows,         // numero di righe
  const ulong  cols,         // numero di colonne
  const int    ndiag=0       // indice della diagonale
   );
```

Parametri

rows

[in] Numero di righe in uscita.

cols

[in] Numero di colonne in uscita.

ndiag=0

[in] Indice della diagonale: 0 (predefinito) si riferisce alla diagonale principale, un valore positivo si riferisce ad una diagonale superiore, e un valore negativo ad una diagonale inferiore.

Valore Restituito

Matrice in cui tutti gli elementi sono uguali a zero, ad eccezione della diagonale k, i cui valori sono uguali a uno.

Esempio MQL5:

```
  matrix eye=matrix::Eye(3, 3);
  Print("eye = \n", eye);

  eye=matrix::Eye(4, 4,1);
  Print("eye = \n", eye);
  /*
   eye =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
   eye =
   [[0,1,0,0]
    [0,0,1,0]
    [0,0,0,1]
    [0,0,0,0]]
  */
```

Esempio Python:

```
np.eye(3, dtype=int)
```

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])


 np.eye(4, k=1)
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.]])
```

Identity

È una funzione statica che crea una matrice di identità della dimensione specificata (non necessariamente quadrata). Una matrice identità contiene tutti uno sulla diagonale principale e zeri altrove. La diagonale principale è costituita dagli elementi della matrice aventi indici di righe e colonne uguali, come [0,0],[1,1],[2,2] ecc. Crea una nuova matrice di identità.

Con il metodo Identity c'è anche la possibilità di trasformare una matrice già esistente in una identità.

```
static matrix matrix::Identity(
  const ulong  rows,         // numero di righe
  const ulong  cols,         // numero di colonne
   );

void matrix::Identity();
```

Parametri

rows

[in] Numero di righe (e colonne) in una matrice n x n.

Valore Restituito

Restituisce la matrice identità. La matrice identità è una matrice quadrata con tutti uno sulla diagonale principale.

Esempio MQL5:

```
  matrix identity=matrix::Identity(3,3);
  Print("identity = \n", identity);
/*
   identity =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
*/
  matrix identity2(3,5);
  identity2.Identity();
  Print("identity2 = \n", identity2);
/*
   identity2 =
   [[1,0,0,0,0]
    [0,1,0,0,0]
    [0,0,1,0,0]]
*/
```

Esempio Python:

```
np.identity(3)
```

```
array([[1.,  0.,  0.],
       [0.,  1.,  0.],
       [0.,  0.,  1.]])
```

Ones

Questa è una funzione statica che crea e restituisce una nuova matrice riempita con tutti uno.

```
static matrix matrix::Ones(
  const ulong  rows,      // numero di righe
  const ulong  cols       // numero di colonne
   );

static vector vector::Ones(
  const ulong  size,      // dimensione del vettore
   );
```

Parametri

rows

[in] Numero di righe.

cols

[in] Numero di colonne.

Valore Restituito

Una nuova matrice di righe e colonne date, riempita con tutti uno.

Esempio MQL5:

```
  matrix ones=matrix::Ones(4, 4);
  Print("ones = \n", ones);
/*
ones =
   [[1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]]
*/
```

Esempio Python:

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

Zeros

Questa è una funzione statica che crea e restituisce una nuova matrice riempita con zeri.

```
static matrix matrix::Zeros(
  const ulong  rows,      // numero di righe
  const ulong  cols       // numero di colonne
   );

static vector vector::Zeros(
  const ulong  size,      // dimensione del vettore
   );
```

Parametri

rows

[in] Numero di righe.

cols

[in] Numero di colonne.

Valore Restituito

Una nuova matrice di righe e colonne date, riempita con zeri.<segmento 0831 >

Esempio MQL5:

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

Esempio Python:

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

Full

La funzione statica crea e restituisce una nuova matrice riempita con un determinato valore.

```
static matrix matrix::Full(
  const ulong  rows,         // numero di righe
  const ulong  cols,         // numero di colonne
  const double  value       // valore con cui riempire
   );

static vector vector::Full(
  const ulong   size,       // dimensione del vettore
  const double  value       // valore con cui riempire
   );
```

Parametri

rows

[in] Numero di righe.

cols

[in] Numero di colonne.

value

[in] Valore con cui riempire tutti gli elementi della matrice

Valore Restituito

Restituisce una nuova matrice di righe e colonne indicate, riempite con il valore specificato.

Esempio MQL5:

```
  matrix full=matrix::Full(3,4,10);
  Print("full = \n", full);
/*
full =
   [[10,10,10,10]
    [10,10,10,10]
    [10,10,10,10]]
*/
```

Esempio Python:

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

Tri

Si tratta di una funzione statica che costruisce una matrice con tutti uno in corrispondenza e sotto la diagonale data e zeri altrove.

```
static matrix matrix::Tri(
  const ulong rows,         // numero di righe
  const ulong cols,         // numero di colonne
  const int   ndiag=0       // indice della diagonale
   );
```

Parametri

rows

[in] Numero di righe nell'array.

cols

[in] Numero di colonne nell'array.

ndiag=0

[in] La sottodiagonale al di sotto della quale l'array è riempito. k = 0 è la diagonale principale, mentre k < 0 è sotto di essa e k > 0 è sopra. Il valore predefinito è 0.

Valore Restituito

Array con il suo triangolo inferiore riempito con uno e zero altrove.

Esempio MQL5:

```
   matrix matrix_a=matrix::Tri(3,4,1);
   Print("Tri(3,4,1)\n",matrix_a);
   matrix_a=matrix::Tri(4,3,-1);
   Print("Tri(4,3,-1)\n",matrix_a);

/*
   Tri(3,4,1)
   [[1,1,0,0]
    [1,1,1,0]
    [1,1,1,1]]
   Tri(4,3,-1)
   [[0,0,0]
    [1,0,0]
    [1,1,0]
    [1,1,1]]
*/
```

Esempio:

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1]])
```

Init

Inizializza una matrice o un vettore.

```
void matrix::Init(
  const ulong  rows,            // numero di righe
  const ulong  cols,            // numero di colonne
  func_name    init_func=NULL,  // funzione init collocata in un ambito o metodo stati
   ...         parameters
   );

void vector::Init(
  const ulong  size,            // dimensione del vettore
  func_name    init_func=NULL,  // funzione init collocata in un ambito o metodo stati
   ...         parameters
   );
```

Parametri

rows

[in] Numero di righe.

cols

[in] Numero di colonne.

func_name

[in] Funzione di Inizializzazione.

...

[in] Parametri della funzione di inizializzazione.

Valore Restituito

Nessun valore restituito.

Esempio:

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
```

```
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
  int size_m=3, size_k=4;
  matrix  m(size_m,size_k,MatrixArange,-2.,0.1); //prima viene creata una matrice non
  Print("matrix m \n",m);                        // poi viene chiamata la funzione Mat
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);   // dopo viene creata una matrice di t
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);          // poi viene creato un vettore, viene
  Print("vector v \n",v);
  /*
   matrix m
   [[-2,-1.9,-1.8,-1.7]
    [-1.6,-1.5,-1.399999999999999,-1.299999999999999]
    [-1.199999999999999,-1.099999999999999,-0.9999999999999992,-0.8999999999999992]]
   matrix m_float
   [[-2,-1.9,-1.8,-1.6999999,-1.5999999]
    [-1.4999999,-1.3999999,-1.2999998,-1.1999998,-1.0999998]
    [-0.99999976,-0.89999974,-0.79999971,-0.69999969,-0.59999967]
    [-0.49999967,-0.39999968,-0.29999968,-0.19999969,-0.099999689]
    [3.1292439e-07,0.10000031,0.20000032,0.30000031,0.4000003]]
   vector v
   [-10,-9,-8,-7]
  */
  }
```

Fill

Riempie una matrice o un vettore esistente con il valore specificato.

```
void matrix::Fill(
  const double  value      // valore con cui riempire
   );

void vector::Fill(
  const double  value      // valore con cui riempire
   );
```

Parametri

value

[in] Valore con cui riempire tutti gli elementi della matrice

Valore Restituito

Nessun valore restituito. La matrice viene riempita con il valore specificato.

Esempio:

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

Manipolazioni di matrici e vettori

Questi sono metodi per le operazioni di matrice di base: riempimento, copia, ottenere una parte di una matrice, trasposizione, divisione e ordinamento.

Esistono anche diversi metodi per le operazioni con righe e colonne di matrice.

Funzione	Azione
HasNan	Restituisce il numero dei valori NaN in una matrice/vettore
Transpose	Inverte o permuta gli assi di una matrice; restituisce la matrice modificata
TriL	Restituisce una copia di una matrice con elementi sopra la diagonale k azzerata. Matrice triangolare inferiore
TriU	Restituisce una copia di una matrice con elementi sotto la diagonale k azzerata. Matrice triangolare superiore
Diag	Estrae una diagonale o costruire la diagonale di una matrice
Row	Restituisce la riga del vettore. Scrivi un vettore nella riga specificata
Col	Restituisce la colonna di un vettore. Scrivi un vettore nella colonna specificata
Copy	Restituisce una copia della matrice/vettore data
Compare	Confronta gli elementi di due matrici/vettori con la precisione specificata
CompareByDigits	Confronta gli elementi di due matrici/vettori fino a cifre significative
Flat	Consente di indirizzare un elemento matrice attraverso un indice invece di due
Clip	Limita gli elementi di una matrice/vettore a un intervallo specificato di valori validi
Reshape	Modifica la forma di una matrice senza modificare i dati
Resize	Restituisce una nuova matrice con forma e dimensione modificate
SwapRows	Scambia le righe in una matrice
SwapCols	Scambia le colonne in una matrice
Split	Divide una matrice in più sottomatrici
Hsplit	Divide una matrice orizzontalmente in più sottomatrici. Uguale a Split con asse = 0
Vsplit	Divide una matrice verticalmente in più sottomatrici. Uguale a Split con asse = 1
ArgSort	Ordina indirettamente una matrice o un vettore.

Funzione	Azione
Sort	Ordina una matrice o un vettore sul posto.

HasNan

Restituisce il numero dei valori NaN in una matrice/vettore.

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

Valore Restituito

Il numero degli elementi della matrice/vettore che contengono un valore NaN.

Nota

Quando si confrontano le appropriate coppie di elementi con valori NaN, i metodi Compare e CompareByDigits considerano questi elementi uguali, mentre nel caso di un normale confronto di numeri in virgola mobile NaN != NaN.

Esempio:

```
void OnStart(void)
  {
   double x=sqrt(-1);

   Print("single: ",x==x);

   vector<double> v1={x};
   vector<double> v2={x};

   Print("vector: ", v1.Compare(v2,0)==0);
  }

/* Result:

 single: false
 vector: true
*/
```

Vedere anche

MathClassify, Compare, CompareByDigits

Transpose

Trasposizione della matrice. Inverte o permuta gli assi di una matrice; restituisce la matrice modificata.

```
matrix matrix::Transpose()
```

Valore Restituito

Matrice trasposta.

Un semplice algoritmo di trasposizione della matrice in MQL5:

```
matrix MatrixTranspose(const matrix& matrix_a)
  {
   matrix matrix_c(matrix_a.Cols(),matrix_a.Rows());

   for(ulong i=0; i<matrix_c.Rows(); i++)
      for(ulong j=0; j<matrix_c.Cols(); j++)
         matrix_c[i][j]=matrix_a[j][i];

   return(matrix_c);
  }
```

Esempio MQL5:

```
  matrix a= {{0, 1, 2}, {3, 4, 5}};
  Print("matrix a \n", a);
  Print("a.Transpose() \n", a.Transpose());

  /*
  matrix a
  [[0,1,2]
   [3,4,5]]
  a.Transpose()
  [[0,3]
   [1,4]
   [2,5]]
  */
```

Esempio Python:

```
import numpy as np

a = np.arange(6).reshape((2,3))
print("a \n",a)
print("np.transpose(a) \n",np.transpose(a))
```

```
a
 [[0 1 2]
 [3 4 5]]
np.transpose(a)
 [[0 3]
 [1 4]
 [2 5]]
```

TriL

Restituisce una copia di una matrice con elementi sopra la diagonale k azzerata. Matrice triangolare inferiore.

```
matrix matrix::Tril(
  const int      ndiag=0      // indice della diagonale
   );
```

Parametri

ndiag=0

[in] Diagonale i cui elementi sopra con zero. ndiag = 0 (default) è la diagonale principale, ndiag < 0 è sotto di essa e ndiag > 0 è sopra.

Valore Restituito

Array con il suo triangolo inferiore riempito con uno e zero altrove.

Esempio MQL5:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriL(-1);
   Print("matrix b \n",b);

  /*
  matrix_c
  [[0,0,0]
  [4,0,0]
  [7,8,0]
  [10,11,12]]
  */
```

Esempio Python:

```
import numpy as np

a=np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)

[[ 0  0  0]
 [ 4  0  0]
 [ 7  8  0]
 [10 11 12]]
```

TriU

Restituisce una copia di una matrice con elementi sotto la diagonale k azzerata. Matrice triangolare superiore.

```
matrix matrix::Triu(
  const int      ndiag=0      // indice della diagonale
   );
```

Parametri

ndiag=0

[in] Diagonale i cui elementi sotto con zero. ndiag = 0 (default) è la diagonale principale, ndiag < 0 è sotto di essa e ndiag > 0 è sopra.

Esempio MQL5:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriU(-1);
   Print("matrix b \n",b);

  /*
  matrix b
  [[1,2,3]
   [4,5,6]
   [0,8,9]
   [0,0,12]]
  */
```

Esempio Python:

```
import numpy as np

a=np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)
print(a)

[[ 1  2  3]
 [ 4  5  6]
 [ 0  8  9]
 [ 0  0 12]]
```

Diag

Estrae una diagonale o costruisce una diagonale di matrice.

```
vector matrix::Diag(
  const int     ndiag=0      // numero della diagonale
   );

void matrix::Diag(
  const vector  v,           // vettore diagonale
  const int     ndiag=0      // numero della diagonale
   );
```

Parametri

v

[in] Un vettore i cui elementi saranno contenuti nella diagonale corrispondente (ndiag=0 è la diagonale principale)

ndiag=0

[in] Diagonale in questione. Predefinita è 0. Usa ndiag>0 per le diagonali sopra la diagonale principale e ndiag<0 per le diagonali sotto la diagonale principale.

Note

È possibile impostare una diagonale per le matrici non assegnate (che non hanno dimensioni). In questo caso, verrà creata una matrice zero della dimensione corrispondente alla dimensione del vettore diagonale, dopodichè i valori del vettore saranno inseriti nella diagonale corrispondente. Se la diagonale è impostata su una matrice già esistente, le dimensioni della matrice non cambiano e i valori degli elementi della matrice al di fuori del vettore diagonale non cambiano.

Esempio

```
   vector v1={1,2,3};
   matrix m1;
   m1.Diag(v1);
   Print("m1\n",m1);
   matrix m2;
   m2.Diag(v1,-1);
   Print("m2\n",m2);
   matrix m3;
   m3.Diag(v1,1);
   Print("m3\n",m3);
   matrix m4=matrix::Full(4,5,9);
   m4.Diag(v1,1);
   Print("m4\n",m4);

   Print("diag -1 - ",m4.Diag(-1));
```

```
  Print("diag 0 - ",m4.Diag());
  Print("diag 1 - ",m4.Diag(1));

/*

m1
[[1,0,0]
[0,2,0]
[0,0,3]]
m2
[[0,0,0]
[1,0,0]
[0,2,0]
[0,0,3]]
m3
[[0,1,0,0]
[0,0,2,0]
[0,0,0,3]]
m4
[[9,1,9,9,9]
[9,9,2,9,9]
[9,9,9,3,9]
[9,9,9,9,9]]
diag -1 - [9,9,9]
diag 0 - [9,9,9,9]
diag 1 - [1,2,3,9]
*/
```

Row

Restituisce la riga del vettore. Scrivi un vettore nella riga specificata

```
vector matrix::Row(
  const ulong   nrow      // numero di riga
   );

void matrix::Row(
  const vector  v,        // riga del vettore
  const ulong   nrow      // numero di riga
   );
```

Parametri

nrow

[in] Numero di riga.

Valore Restituito

Vettore.

Note

È possibile impostare una riga per le matrici non allocate (che non hanno dimensioni). In questo caso, verrà creata una matrice zero della dimensione del numero di riga x + 1 del vettore, dopo di che i valori degli elementi vettoriali verranno inseriti nella riga corrispondente. Se la riga è impostata su una matrice già esistente, le dimensioni della matrice non cambiano e i valori degli elementi della matrice al di fuori della riga del vettore non cambiano.

Esempio

```
   vector v1={1,2,3};
   matrix m1;
   m1.Row(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,7);
   m2.Row(v1,2);
   Print("m2\n",m2);

   Print("row 1 - ",m2.Row(1));
   Print("row 2 - ",m2.Row(2));

  /*
  m1
  [[0,0,0]
  [1,2,3]]
  m2
```

```
   [[7,7,7,7,7]
   [7,7,7,7,7]
   [1,2,3,7,7]
   [7,7,7,7,7]]
   row 1 - [7,7,7,7,7]
   row 2 - [1,2,3,7,7]
   */
```

Col

Restituisce la colonna di un vettore. Scrivi un vettore nella colonna specificata.

```
vector matrix::Col(
  const ulong   ncol      // numero della colonna
   );

void matrix::Col(
  const vector  v,        // colonna del vettore
  const ulong   ncol      // numero della colonna
   );
```

Parametri

ncol

[in] Numero della colonna.

Valore Restituito

Vettore.

Note

È possibile impostare una colonna per le matrici non assegnate (che non hanno dimensioni). In questo caso, verrà creata una matrice zero con la dimensione del vettore con la dimensione del numero della colonna + 1, dopo di che i valori degli elementi vettoriali saranno inseriti nella colonna corrispondente. Se la colonna è impostata su una matrice già esistente, le dimensioni della matrice non cambiano e i valori degli elementi della matrice al di fuori della colonna del vettore non cambiano.

Esempio

```
   vector v1={1,2,3};
   matrix m1;
   m1.Col(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,8);
   m2.Col(v1,2);
   Print("m2\n",m2);

   Print("col 1 - ",m2.Col(1));
   Print("col 2 - ",m2.Col(2));

  /*
  m1
  [[0,1]
  [0,2]
  [0,3]]
```

```
m2
[[8,8,1,8,8]
[8,8,2,8,8]
[8,8,3,8,8]
[8,8,8,8,8]]
col 1 - [8,8,8,8]
col 2 - [1,2,3,8]
*/
```

Copy

Crea una copia della matrice/vettore data.

```
bool matrix::Copy(
  const matrix&  a     // matrice copiata
   );
bool vector::Copy(
  const vector&  v     // vettore copiato
   );
```

Parametri

v

[in] Matrice o vettore da copiare.

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Esempio MQL5:

```
  matrix a=matrix::Eye(3, 4);
  matrix b;
  b.Copy(a);
  matrix c=a;
  Print("matrix b \n", b);
  Print("matrix_c \n", c);

  /*
  /*
  matrix b
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  matrix_c
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  */
  */
```

Esempio Python:

```
import numpy as np

a = np.eye(3,4)
print('a \n',a)
b = a
```

```
print('b \n',b)
c = np.copy(a)
print('c \n',c)

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
b
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
c
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
```

Compare

Confronta gli elementi di due matrici/vettori con la precisione specificata.

```
ulong vector::Compare(
  const vector& vec,          // vettore da confrontare
  const double  epsilon       // precisione
   );

ulong matrix::Compare(
  const matrix& mat,          // matrice da confrontare
  const double  epsilon       // precisione
   );
```

Parametri

vector_b

[in] Vettore da confrontare.

epsilon

[in] Precisione.

Valore Restituito

Numero di elementi non corrispondenti delle matrici o dei vettori confrontati: 0 se le matrici sono uguali, maggiore di 0 altrimenti.

Note

Gli operatori di confronto == o != eseguono un confronto esatto. È noto che il confronto esatto dei numeri reali è di uso limitato, quindi è stato aggiunto il metodo di confronto epsilon. Può accadere che una matrice possa contenere elementi in un intervallo, ad esempio da 1e-20 a 1e+20. Tali matrici possono essere elaborate utilizzando il confronto degli elementi fino a cifre significative.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4}};
   matrix matrix_i=matrix::Identity(3,3);
   matrix matrix_c=matrix_a.Inv();
   matrix matrix_check=matrix_a.MatMul(matrix_c);
   Print("matrix_check\n",matrix_check);

   ulong errors=matrix_check.Compare(matrix::Identity(3,3),1e-15);
   Print("errors=",errors);


  /*
```

```
matrix_check
[[1,0,0]
[4.440892098500626e-16,1,8.881784197001252e-16]
[4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
errors=0

*/
```

CompareByDigits

Confrontare gli elementi di due matrici/ vettori con la precisione delle cifre significative.

```
ulong vector::CompareByDigits(
  const vector& vec,          // vettore da confrontare
  const int     digits        // numero di cifre significative
   );

ulong matrix::CompareByDigits(
  const matrix& mat,          // matrice da confrontare
  const int     digits        // numero di cifre significative
   );
```

Parametri

vector_b

[in] Vettore da confrontare.

digits

[in] Numero di cifre significative da confrontare.

epsilon

[in] Precisione di confronto. Se due valori differiscono in valore assoluto di meno della precisione specificata, sono considerati uguali.

Valore Restituito

Numero di elementi non corrispondenti delle matrici o dei vettori confrontati: 0 se le matrici sono uguali, maggiore di 0 altrimenti.

Note

Gli operatori di confronto == o != eseguono un confronto esatto. È noto che il confronto esatto dei numeri reali è di uso limitato, quindi è stato aggiunto il metodo di confronto epsilon. Può accadere che una matrice possa contenere elementi in un intervallo, ad esempio da 1e-20 a 1e+20. Tali matrici possono essere elaborate utilizzando il confronto degli elementi fino a cifre significative.

Esempio

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//-- riempire la matrice
   double  value=0.0;
   for(int i=0; i<size_m; i++)
     {
      for(int j=0; j<size_k; j++)
        {
```

```
         if(i==j)
            matrix_a[i][j]=1.0+i;
         else
           {
            value+=1.0;
            matrix_a[i][j]=value/1e+20;
           }
        }
     }
//-- ottenere un'altra matrice
   matrix matrix_c = matrix_a * -1;

   ulong errors_epsilon=matrix_a.Compare(matrix_c,1e-15);
   ulong errors_digits=matrix_a.CompareByDigits(matrix_c,15);

   printf("Compare matrix %d x %d  errors_epsilon=%I64u  errors_digits=%I64u",size_m,s


  /*
  Compare matrix 128 x 256  errors_epsilon=128  errors_digits=32768
  */
```

Flat

Consente di indirizzare un elemento matrice attraverso un indice invece di due.

```
bool matrix::Flat(
  const ulong   index,      //
  const double  value       // valore da impostare
   );

double matrix::Flat(
  const ulong   index,      //
   );
```

Parametri

index

[in] Indice Flat

value

[in] Valore da impostare tramite l'indice dato.

Valore Restituito

Valore tramite indice dato.

Note

Per la matrice mat(3,3), l'accesso può essere scritto come segue:

- lettura: 'x=mat.Flat(4)', che è equivalente a 'x=mat[1][1]'
- scrittura: 'mat.Flat(5, 42)', equivalente a 'mat[1][2]=42'

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
   ulong arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));
   matrix_a.Flat(arg_max,0);
   arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));


  /*
  matrix_a
  [[10,3,2]
   [1,8,12]
   [6,5,4]
   [7,11,9]]
```

```
max_value=12.0
max_value=11.0
*/
```

Clip

Limita gli elementi di una matrice/vettore a un intervallo specificato di valori validi.

```
bool matrix::Clip(
  const double  min_value,     // valore minimo
  const double  max_value      // valore massimo
   );
bool vector::Clip(
  const double  min_value,     // valore minimo
  const double  max_value      // valore massimo
   );
```

Parametri

min_value

[in] Valore minimo.

max_value

[in] Valore massimo.

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Note

La matrice (o vettore) viene elaborata sul posto. Non vengono create copie.

Esempio

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   bool res=matrix_a.Clip(4,8);
   Print("matrix_a\n",matrix_a);

  /*
  matrix_a
  [[4,4,4]
   [4,5,6]
   [7,8,8]
   [8,8,8]]
  */
```

Reshape

Modificare la forma di una matrice senza modificare i suoi dati.

```
void  Reshape(
  const ulong  rows,      // nuovo numero di righe.
  const ulong  cols       // nuovo numero di colonne.
   );
```

Parametri

rows

[in] Nuovo numero di righe.

cols

[in] Nuovo numero di colonne.

Note

La matrice viene elaborata sul posto. Non vengono create copie. È possibile specificare qualsiasi dimensione, ad es., rows_new*cols_new!=rows_old*cols_old. Quando il buffer della matrice viene incrementato, i valori extra non sono definiti.

Esempio

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

   Print("matrix_a\n",matrix_a);
   matrix_a.Reshape(2,6);
   Print("Reshape(2,6)\n",matrix_a);
   matrix_a.Reshape(3,5);
   Print("Reshape(3,5)\n",matrix_a);
   matrix_a.Reshape(2,4);
   Print("Reshape(2,4)\n",matrix_a);

  /*
  matrix_a
  [[1,2,3]
   [4,5,6]
   [7,8,9]
   [10,11,12]]
  Reshape(2,6)
  [[1,2,3,4,5,6]
   [7,8,9,10,11,12]]
  Reshape(3,5)
  [[1,2,3,4,5]
   [6,7,8,9,10]
   [11,12,0,3,0]]
  Reshape(2,4)
  [[1,2,3,4]
```

```
   [5,6,7,8]]
  */
```

Resize

Restituisce una nuova matrice con forma e dimensione modificate.

```
bool matrix::Resize(
  const ulong  rows,     //
  const ulong  cols,     // nuovo numero di colonne.
  const ulong  reserve=0 // quantità di riserva in elementi.
   );

bool vector::Resize(
  const ulong  size,     // nuova dimensione.
  const ulong  reserve=0 // quantità di riserva in elementi.
   );
```

Parametri

rows

[in] Nuovo numero di righe.

cols

[in] Nuovo numero di colonne.

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Note

La matrice (o vettore) viene elaborata sul posto. Non vengono create copie. È possibile specificare qualsiasi dimensione, ad es., rows_new*cols_new!=rows_old*cols_old. A differenza di Reshape, la matrice viene elaborata riga per riga. Quando si aumenta il numero di colonne, i valori delle colonne extra non sono definiti. Quando si aumenta il numero di righe, i valori degli elementi nelle nuove righe non sono definiti. Quando il numero di colonne viene ridotto, ogni riga della matrice viene troncata.

Esempio

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);
   matrix_a.Resize(2,6);
   Print("Ressize(2,6)\n",matrix_a);
   matrix_a.Resize(3,5);
   Print("Resize(3,5)\n",matrix_a);
   matrix_a.Resize(2,4);
   Print("Resize(2,4)\n",matrix_a);

   /*
   matrix_a
```

```
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   Ressize(2,6)
   [[1,2,3,4,5,6]
    [4,5,6,10,11,12]]
   Resize(3,5)
   [[1,2,3,4,5]
    [4,5,6,10,11]
    [11,12,3,8,8]]
   Resize(2,4)
   [[1,2,3,4]
    [4,5,6,10]]
   */
```

Set

Imposta il valore di un elemento del vettore in base all'indice specificato.

```
bool vector::Set(
  ulong   index,     // indice elemento
  double  value      // valore
   );
```

Parametri

index

[in] Indice dell'elemento per cui deve essere impostato il valore.

value

[in] Valore.

Valore Restituito

Restituisce true in caso di successo, altrimenti - false.

Nota

Il metodo Set fa la stessa cosa di assegnare un valore utilizzando parentesi quadre, vale a dire: *vector[indice]=valore* .Il metodo è stato aggiunto per semplificare il trasferimento di un codice tra i linguaggi in cui viene utilizzato questo tipo di notazione. L'esempio seguente mostra entrambe le opzioni per riempire il vettore con valori tramite l'indice specificato.

Esempio:

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
 /* Risultato
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------+
//| Riempire un vettore con potenze di un numero tramite operazione di assegnazione |
//+------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
```

```
   for(ulong k=0; k<v.Size(); k++)
     {
      v[k]=value;
      value*=2;
     }
  }
//+------------------------------------------------------------------------------+
//| Riempire un vettore con potenze di un numero utilizzando il metodo Set |
//+------------------------------------------------------------------------------+
void VectorSetValues(vector& v, double initial=1)
  {
   double value=initial;
   for(ulong k=0; k<v.Size(); k++)
     {
      v.Set(k, value);
      value*=2;
     }
  }
```

SwapRows

Scambia le righe in una matrice.

```
bool matrix::SwapRows(
  const ulong  row1,     // indice della prima riga
  const ulong  row2      // indice della seconda riga
   );
```

Parametri

row1

[in] Indice della prima riga.

row2

[in] Indice della seconda riga.

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Esempio

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapRows(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapRows(0,3);
   matrix matrix_c1=matrix_p.MatMul(matrix_a);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  matrix_c1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  */
```

SwapCols

Scambia le colonne in una matrice

```
bool matrix::SwapCols(
  const ulong  row1,     // indice della prima colonna
  const ulong  row2      // indice della seconda colonna
   );
```

Parametri

col1

[in] Indice della prima colonna.

col2

[in] Indice della seconda colonna

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Esempio

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapCols(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapCols(0,3);
   matrix matrix_c1=matrix_a.MatMul(matrix_p);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  matrix_c1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  */
```

Split

Divide una matrice in più sottomatrici.

```
bool matrix::Split(
  const ulong  parts,      // numero di sottoatrici
  const int    axis,       // asse
  matrix&      splitted[]  // array delle sottomatrici risultanti
   );

void matrix::Split(
  const ulong& parts[],    // dimensioni delle sottomatrici
  const int    axis,       // asse
  matrix&      splitted[]  // array delle sottomatrici risultanti
   );
```

Parametri

parts

[in] Il numero di sottomatrici in cui dividere la matrice.

axis

[in] Asse. 0 - asse orizzontale, 1 - asse verticale.

splitted

[out] Array delle sottomatrici risultanti.

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Note

Se viene specificato il numero di sottomatrici, si ottengono submatrici con le stesse dimensioni. Significa che la dimensione della matrice (0 - il numero di righe, 1 - il numero di colonne) deve essere divisibile per le 'parti' senza resto. È possibile ottenere sottomatrici di dimensioni diverse utilizzando un array delle dimensioni delle sottomatrici. Gli elementi di dimensioni dell'array vengono utilizzati fino a quando l'intera matrice viene divisa. Se la dimensione dell'array è terminato e la matrice non è ancora stata completamente divisa, il resto indiviso sarà l'ultima sottomatrice.

Esempio

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
```

```
   ulong  parts[]={2,2};

   bool res=matrix_a.Split(2,0,splitted);
   Print(res,"  ",GetLastError());
   ResetLastError();
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Split(2,1,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Split(parts,0,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

  /*
  false  4003
  true  0
  splitted 0
  [[1,2,3]
   [7,8,9]
   [13,14,15]
   [19,20,21]
   [25,26,27]]
  splitted 1
  [[4,5,6]
   [10,11,12]
   [16,17,18]
   [22,23,24]
   [28,29,30]]
  true  0
  splitted 0
  [[1,2,3,4,5,6]
   [7,8,9,10,11,12]]
  splitted 1
  [[13,14,15,16,17,18]
   [19,20,21,22,23,24]]
  splitted 2
  [[25,26,27,28,29,30]]
  */
```

Hsplit

Divide una matrice orizzontalmente in più sottomatrici. Uguale a Split con asse = 0

```
bool matrix::Hsplit(
  const ulong  parts,      // numero di sottoatrici
  matrix&      splitted[]  // array delle sottomatrici risultanti
   );

void matrix::Hsplit(
  const ulong& parts[],    // dimensioni delle sottomatrici
  matrix&      splitted[]  // array delle sottomatrici risultanti
   );
```

Parametri

parts

[in] Il numero di sottomatrici in cui dividere la matrice.

splitted

[out] Array delle sottomatrici risultanti.

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Note

Se viene specificato il numero di sottomatrici, si ottengono submatrici con le stesse dimensioni. Ciò significa che il numero di righe deve essere divisibile per le 'parti' senza resto. È possibile ottenere sottomatrici di dimensioni diverse utilizzando un array delle dimensioni delle sottomatrici. Gli elementi di dimensioni dell'array vengono utilizzati fino a quando l'intera matrice viene divisa. Se la dimensione dell'array è terminato e la matrice non è ancora stata completamente divisa, il resto indiviso sarà l'ultima sottomatrice.

Esempio

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,4};

   bool res=matrix_a.Hsplit(2,splitted);
   Print(res,"  ",GetLastError());
   ResetLastError();
   for(uint i=0; i<splitted.Size(); i++)
```

```
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(5,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(parts,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);


  /*
  false  4003
  true  0
  splitted 0
  [[1,2,3,4,5,6]]
  splitted 1
  [[7,8,9,10,11,12]]
  splitted 2
  [[13,14,15,16,17,18]]
  splitted 3
  [[19,20,21,22,23,24]]
  splitted 4
  [[25,26,27,28,29,30]]
  true  0
  splitted 0
  [[1,2,3,4,5,6]
  [7,8,9,10,11,12]]
  splitted 1
  [[13,14,15,16,17,18]
  [19,20,21,22,23,24]
  [25,26,27,28,29,30]]
  */
```

Vsplit

Divide una matrice verticalmente in più sottomatrici. Uguale a Split con asse = 1

```
bool matrix::Vsplit(
  const ulong  parts,      // numero di sottoatrici
  matrix&      splitted[]  // array delle sottomatrici risultanti
   );

void matrix::Vsplit(
  const ulong& parts[],    // dimensioni delle sottomatrici
  matrix&      splitted[]  // array delle sottomatrici risultanti
   );
```

Parametri

parts

[in] Il numero di sottomatrici in cui dividere la matrice.

splitted

[out] Array delle sottomatrici risultanti.

Valore Restituito

Restituisce true in caso di successo, altrimenti false.

Note

Se viene specificato il numero di sottomatrici, si ottengono submatrici con le stesse dimensioni. Ciò significa che il numero delle colonne deve essere divisibile per le 'parti' senza resto. È possibile ottenere sottomatrici di dimensioni diverse utilizzando un array delle dimensioni delle sottomatrici. Gli elementi di dimensioni dell'array vengono utilizzati fino a quando l'intera matrice viene divisa. Se la dimensione dell'array è terminato e la matrice non è ancora stata completamente divisa, il resto indiviso sarà l'ultima sottomatrice.

Esempio

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18}};
   matrix splitted[];
   ulong  parts[]={2,3};

   matrix_a.Vsplit(2,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(3,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);
```

```
 matrix_a.Vsplit(parts,splitted);
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);


/*
   splitted 0
   [[1,2,3]
    [7,8,9]
    [13,14,15]]
   splitted 1
   [[4,5,6]
    [10,11,12]
    [16,17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4]
    [9,10]
    [15,16]]
   splitted 2
   [[5,6]
    [11,12]
    [17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4,5]
    [9,10,11]
    [15,16,17]]
   splitted 2
   [[6]
    [12]
    [18]]

*/
```

ArgSort

Ordinamento indiretto di matrice o vettore.

```
vector vector::Sort(
  func_name  compare_func=NULL,  // funzione di confronto
  T          context             // parametro per la funzione di ordinamento personali
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // funzione di confronto
  T          context             // parametro per la funzione di ordinamento personali
   );

matrix matrix::Sort(
  const int  axis,               // asse per l'ordinamento
  func_name  compare_func=NULL   // funzione di confronto
  T          context             // parametro per la funzione di ordinamento personali
   );
```

Parametri

axis

[in] L'asse lungo il quale ordinare: 0 è orizzontale, 1 è verticale.

func_name

[in] Comparatore. È possibile specificare uno dei valori dell'enumerazione ENUM_SORT_MODE o della propria funzione di confronto. Se non viene specificata alcuna funzione, viene utilizzato l'ordinamento ascendente.

Una funzione di confronto personalizzata può essere di due tipi:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Qui T è il tipo di matrice o vettore, e TContex è il tipo della variabile 'context' che viene passata come parametro aggiuntivo al metodo Sort.

context

[in] Parametro opzionale aggiuntivo che può essere passato ad una funzione di ordinamento personalizzata.

Valore Restituito

Vettore o matrice con gli indici degli elementi ordinati. Per esempio, il risultato [4,2,0,1,3] indica che ci dovrebbe essere un elemento con indice 4 nella posizione zero, un elemento con indice 2 nella prima posizione, e così via.

Sort

Ordina una matrice o un vettore sul posto.

```
void vector::Sort(
  func_name  compare_func=NULL,  // funzione di confronto
  T          context             // parametro per la funzione di ordinamento personali
   );

void matrix::Sort(
  func_name  compare_func=NULL   // funzione di confronto
  T          context             // parametro per la funzione di ordinamento personali
   );

void matrix::Sort(
  const int  axis,               // asse per l'ordinamento
  func_name  compare_func=NULL   // funzione di confronto
  T          context             // parametro per la funzione di ordinamento personali
   );
```

Parametri

axis

[in] L'asse lungo il quale ordinare: 0 è orizzontale, 1 è verticale.

func_name

[in] Comparatore. È possibile specificare uno dei valori dell'enumerazione ENUM_SORT_MODE o della propria funzione di confronto. Se non viene specificata alcuna funzione, viene utilizzato l'ordinamento ascendente.

Una funzione di confronto personalizzata può essere di due tipi:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Qui T è il tipo di matrice o vettore, e TContex è il tipo della variabile 'context' che viene passata come parametro aggiuntivo al metodo Sort.

context

[in] Parametro opzionale aggiuntivo che può essere passato ad una funzione di ordinamento personalizzata.

Valore Restituito

Nessuno. L'ordinamento viene eseguito sul posto, cioè viene applicato ai dati della matrice/vettore per il quale viene chiamato il metodo Sort.

Esempio

```
//+------------------------------------------------------------------+
//| Sort function                                                    |
//+------------------------------------------------------------------+
```

```
int MyDoubleComparator(double x1,double x2,int sort_mode=0)
  {
   int res=x1<x2 ? -1 : (x1>x2 ? 1 : 0);
   return(sort_mode==0 ? res : -res);
  }
//+------------------------------------------------------------------+
//| Script start function                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- riempire il vettore
   vector v(100);
   //--- ordinamento ascendente
   v.Sort(MyDoubleComparator);   // viene utilizzato un parametro aggiuntivo con il va
   Print(v);
   // ordinamento decrescente
   v.Sort(MyDoubleComparator,1); //qui il parametro aggiuntivo '1' è esplicitamente sp
   Print(v);
  }
```

Operazioni matematiche con matrici e vettori

Le operazioni matematiche, tra cui addizione, sottrazione, moltiplicazione e divisione, possono essere eseguite sugli opportuni elementi di matrici e vettori.

Le funzioni matematiche erano originariamente progettate per eseguire operazioni rilevanti su valori scalari. La maggior parte delle funzioni possono essere applicate a matrici e vettori. Queste includono MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, and MathTanh. Tali operazioni implicano l'elaborazione di opportuni elementi di matrici e vettori. Esempio

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

Per MathMod e MathPow, il secondo elemento può essere sia uno scalare o una matrice/vettore della dimensione appropriata.

L'esempio seguente mostra come calcolare la deviazione standard applicando funzioni matematiche a un vettore.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//-- usa la funzione di inizializzazione per riempire il vettore
  vector r(10, ArrayRandom); // array di numeri casuali da 0 a 1
//-- calcolare il valore medio
  double avr=r.Mean();       // array valore medio
  vector d=r-avr;            // calcolare un array di deviazioni dalla media
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // array di deviazioni al quadrato
  double sum=s2.Sum();       // somma delle deviazioni al quadrato
//-- calcolare la deviazione standard in due metodi diversi
  double std=MathSqrt(sum/r.Size());
  Print(" std(r)=", std);
```

```
   Print("r.Std()=", r.Std());
 }
/*
   avr(r)=0.5300302133243813
   r=[0.8346201971495713,0.8031556138798182,0.6696676534318063,0.05386516922513505,0.54
   d=[0.30458998382519,0.2731254005554369,0.1396374401074251,-0.4761650440992462,0.0190
    std(r)=0.2838269732183663
   r.Std()=0.2838269732183663
*/
//+------------------------------------------------------------------+
//| Riempie un vettore con valori casuali                            |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
 {
  for(ulong i=0; i<v.Size(); i++)
    v[i]=double(MathRand())/32767.;
 }
```

Operazioni matematiche

Le operazioni matematiche, tra cui addizione, sottrazione, moltiplicazione e divisione, possono essere eseguite sugli opportuni elementi di matrici e vettori.

Entrambe le matrici o entrambi i vettori devono essere dello stesso tipo e devono avere le stesse dimensioni. Ogni elemento della matrice opera sull'elemento corrispondente della seconda matrice.

È anche possibile utilizzare uno scalare del tipo appropriato (double, float o complex) come secondo termine (moltiplicatore, sottraendo o divisore). In questo caso, ogni membro della matrice o vettore opererà sullo scalare specificato.

```
   matrix matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
   matrix matrix_b={{1,2,3},{4,5,6}};

   matrix matrix_c1=matrix_a+matrix_b;
   matrix matrix_c2=matrix_b-matrix_a;
   matrix matrix_c3=matrix_a*matrix_b;    // Prodotto Hadamard, da non confondere con il
   matrix matrix_c4=matrix_b/matrix_a;

   matrix_c1=matrix_a+1;
   matrix_c2=matrix_b-double_value;
   matrix_c3=matrix_a*M_PI;
   matrix_c4=matrix_b/0.1;

//-- sono possibili operazioni in essere
   matrix_a+=matrix_b;
   matrix_a/=2;
```

Le stesse operazioni sono disponibili per i vettori.

Funzioni matematiche

Le seguenti funzioni matematiche possono essere applicate a matrici e vettori: MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. Tali operazioni implicano l'elaborazione di opportuni elementi di matrici e vettori.

Per MathMod e MathPow, il secondo elemento può essere sia uno scalare o una matrice/vettore della dimensione appropriata.

```
   matrix<T> mat1(128,128);
   matrix<T> mat3(mat1.Rows(),mat1.Cols());
   ulong     n,size=mat1.Rows()*mat1.Cols();
...
   mat2=MathPow(mat1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),(T)1.9);
      if(res!=mat2.Flat(n))
         errors++;
     }
   mat2=MathPow(mat1,mat3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),mat3.Flat(n));
      if(res!=mat2.Flat(n))
         errors++;
     }
...
   vector<T> vec1(16384);
   vector<T> vec3(vec1.Size());
   ulong     n,size=vec1.Size();
...
   vec2=MathPow(vec1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],(T)1.9);
      if(res!=vec2[n])
         errors++;
     }
   vec2=MathPow(vec1,vec3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],vec3[n]);
      if(res!=vec2[n])
         errors++;
     }
```

Prodotti di matrice e vettore

I calcoli del prodotto della matrice e del vettore includono:

- moltiplicazioni di matrici
- moltiplicazioni dii vettori
- calcolo della matrice di covarianza
- calcolo della correlazione incrociata di due vettori
- calcolo della convoluzione di due vettori
- Calcolo del coefficiente di correlazione

Funzione	Azione
MatMul	Prodotto matriciale di due matrici
GeMM	Moltiplicazione generale di matrici (GeMM)
Power	Eleva una matrice quadrata alla potenza intera
Dot	Prodotto scalare di due vettori
Kron	Restituisce il prodotto di Kronecker di due matrici, matrice e vettore, vettore e matrice o due vettori
Inner	Prodotto interno di due matrici
Outer	Calcola il prodotto esterno di due matrici o due vettori
CorrCoef	Calcola il coefficiente di correlazione di Pearson (coefficiente di correlazione lineare)
Cov	Calcola la matrice di covarianza
Correlate	Calcola la correlazione incrociata di due vettori
Convolve	Restituisce la convoluzione lineare discreta di due vettori

MatMul

Il metodo MatMul, che consente la moltiplicazione di matrici e vettori, ha diversi sovraccarichi.

Moltiplicare una matrice per una matrice: `matrix[M][K] * matrix[K][N] = matrix[M][N]`

```
matrix  matrix::MatMul(
  const matrix&  b      // seconda matrice
   );
```

Moltiplicare un vettore per una matrice: `orizzontale vector[K] * matrix[K][N] = orizzontale vector[N]`

```
vector  vector::MatMul(
  const matrix&  b      // matrice
   );
```

Moltiplicare una matrice per un vettore: `matrix[M][K] * verticale vector[K] = verticale vector[M]`

```
vector  matrix::MatMul(
  const vector&  b      // vettore
   );
```

Moltiplicazione vettoriale scalare: `orizzontale vector * verticale vector = valore scalare`

```
scalar  vector::MatMul(
  const vector&  b      // secondo vettore
   );
```

Parametri

b

[in] Matrice o vettore.

Valore Restituito

Matrice, vettore o scalare, a seconda del metodo utilizzato.

Note

Le matrici devono essere compatibili per la moltiplicazione, cioè il numero di colonne nella prima matrice deve essere uguale al numero di righe nella seconda matrice. La moltiplicazione di matrice non è commutativa: il risultato della moltiplicazione della prima matrice per la seconda non è uguale al risultato della moltiplicazione della seconda matrice per la prima nel caso generale.

Il prodotto della matrice è costituito da tutte le possibili combinazioni di prodotti scalari delle righe dei vettori della prima matrice e delle colonne dei vettori della seconda matrice.

Nella moltiplicazione scalare, i vettori devono avere la stessa lunghezza.

Quando si moltiplicano un vettore e una matrice, la lunghezza del vettore deve corrispondere esattamente al numero di colonne nella matrice.

Semplice algoritmo di moltiplicazione della matrice in MQL5:

```
matrix MatrixProduct(const matrix& matrix_a, const matrix& matrix_b)
  {
   matrix matrix_c;

   if(matrix_a.Cols()!=matrix_b.Rows())
      return(matrix_c);

   ulong M=matrix_a.Rows();
   ulong K=matrix_a.Cols();
   ulong N=matrix_b.Cols();
   matrix_c=matrix::Zeros(M,N);

   for(ulong m=0; m<M; m++)
      for(ulong k=0; k<K; k++)
         for(ulong n=0; n<N; n++)
            matrix_c[m][n]+=matrix_a[m][k]*matrix_b[k][n];

   return(matrix_c);
  }
```

Esempio di moltiplicazione di matrice

```
   matrix a={{1, 0, 0},
             {0, 1, 0}};
   matrix b={{4, 1},
             {2, 2},
             {1, 3}};
   matrix c1=a.MatMul(b);
   matrix c2=b.MatMul(a);
   Print("c1 = \n", c1);
   Print("c2 = \n", c2);
/*
   c1 =
   [[4,1]
    [2,2]]
   c2 =
   [[4,1,0]
    [2,2,0]
    [1,3,0]]
*/
```

Un esempio di moltiplicazione di un vettore orizzontale per una matrice

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- creare una matrice 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   vector v3 = {1, 2, 3};
   Print("Product of horizontal vector v and matrix m[3,5]");
   Print("On the left, vector v3 = ", v3);
   Print("On the right, matrix m35 = \n", m35);
   Print("v3.MatMul(m35) = horizontal vector v[5] \n", v3.MatMul(m35));

  /* Result
    Product of horizontal vector v3 and matrix m[3,5]
    On the left, vector v3 = [1,2,3]
    On the right, matrix m35 =
    [[0,1,2,3,4]
     [5,6,7,8,9]
     [10,11,12,13,14]]
    v3.MatMul(m35) = horizontal vector v[5]
    [40,46,52,58,64]
   */
  }
//+------------------------------------------------------------------+
// Riempire la matrice con valori crescenti                          |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

Un esempio di come moltiplicare una matrice per un vettore verticale

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- creare una matrice 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   Print("Product of matrix m[3,5] and vertical vector v[5]");
   vector v5 = {1,2,3,4,5};
   Print("On the left, m35 = \n",m35);
   Print("On the right v5 = ",v5);
   Print("m35.MatMul(v5) = vertical vector v[3] \n",m35.MatMul(v5));

  /* Result
   Product of matrix m[3,5] and vertical vector v[5]
   On the left, m35 =
   [[0,1,2,3,4]
    [5,6,7,8,9]
    [10,11,12,13,14]]
   On the right, v5 = [1,2,3,4,5]
   m35.MatMul(v5) = vertical vector v[3]
   [40,115,190]
   */
  }
//+------------------------------------------------------------------+
// Riempire la matrice con valori crescenti                          |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

Un esempio di prodotto scalare (dot) dei vettori

```
void OnStart()
  {
//-- prodotto scalare di un vettore orizzontale e di un vettore verticale
   vector a= {1, 2, 3};  // vettore orizzontale
   vector b= {4, 5, 6};  // vettore verticale
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
   //--- notare che il metodo Dot genera lo stesso risultato
   Print("2) a.Dot(b) = ", a.Dot(b));

  /* Result
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

Vedere anche

Dot, GeMM

GeMM

Il metodo GeMM (General Matrix Multiply) implementa la moltiplicazione generale di due matrici. L'operazione è definita come C ← α A B + β C , dove le matrici A e B possono essere opzionalmente trasposte . Con una normale moltiplicazione delle matrici AB (MatMul), lo scalare *alpha* si presume sia pari a 1 e *beta* uguale a zero.

La differenza principale tra GeMM e MatMul in termini di efficienza è che MatMul crea sempre una nuova matrice/oggetto vettoriale, mentre GeMM lavora con un oggetto matrice esistente e non lo ricrea. Pertanto, quando si utilizza GeMM la memoria è pre-allocata per la matrice corrispondente, quindi, mentre si lavora con le stesse dimensioni della matrice, non ci saranno riallocazioni della memoria. Questo può essere un vantaggio importante di GeMM per il bulk computing, ad esempio, quando si eseguono ottimizzazioni in un tester di strategia o quando si allena una rete neurale.

Simile a MatMul, GeMM ha anche 4 sovraccarichi. Tuttavia, la semantica del quarto sovraccarico è stata modificata per consentire la moltiplicazione di un vettore verticale con uno orizzontale.

In un oggetto matrice/vettore esistente, non è necessario pre-allocare la memoria. La memoria verrà allocata e riempita di zeri alla prima chiamata GeMM.

Multiplying a matrix by a matrix: `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + β * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // prima matrice
  const matrix &B,     // seconda matrice
  double alpha,        // moltiplicatore alfa per prodotto AB
  double beta,         // moltiplicatore beta per matrice C
  uint   flags         // una combinazione di valori ENUM_GEMM (OR bitwise), che deterr
   );
```

Multiplying a vector by a matrix: `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + β * vector C[N]`

```
bool  vector::GeMM(
  const vector &A,     // vettore orizzontale
  const matrix &B,     // matrice
  double alpha,        // moltiplicatore alfa per prodotto AB
  double beta,         // moltiplicatore beta per vettore C
  uint   flags         // valore di enumerazione ENUM_GEMM che determina se la matrice
   );
```

Multiplying a matrix by a vector: `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + β * vector C[M]`

```
bool  vector::GeMM(
  const matrix &A,     // matrice
  const vector &B,     // vettore verticale
  double alpha,        // moltiplicatore alfa per prodotto AB
  double beta,         // moltiplicatore beta per vettore C
  uint   flags         // Valore di enumerazione ENUM_GEMM che determina se la matrice
   );
```

Moltiplicare un vettore per un vettore: `matrix C[M][N] = α * ( vector A[M] * vector B[N] * ) + β * matrix C[M][N].` Questo sovraccarico restituisce una matrice, a differenza di MatMul dove restituisce uno scalare.

```
bool  matrix::GeMM(
  const vector &A,     // primo vettore
  const vector &B,     // secondo vettore
  double alpha,        // moltiplicatore alfa per prodotto AB
  double beta,         // moltiplicatore beta per matrice C
  uint   flags         // Valore di enumerazione ENUM_GEMM che determina se la matrice
   );
```

Parametri

A

[in] Matrice o vettore.

B

[in] Matrice o vettore.

alpha

[in] Moltiplicatore alfa per il prodotto AB.

beta

[in] Moltiplicatore beta per la matrice C risultante.

flags

[in] Valore di enumerazione ENUM_GEMM che determina se le matrici A, B e C sono trasposte.

Valore Restituito

Restituisce true in caso di successo false altrimenti.

ENUM_GEMM

Enumerazione dei flag per il metodo GeMM.

ID	Descrizione
TRANSP_A	Usa la matrice trasposta A
TRANSP_B	Usa la matrice trasposta B
TRANSP_C	Usa la matrice trasposta C

Note

Matrici e vettori di tipo float, double e complex possono essere utilizzati come parametri A e B. Le varianti dei template del metodo GeMM sono le seguenti:

```
bool matrix<T>::GeMM(const matrix<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
```

```
bool matrix<T>::GeMM(const vector<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)

bool vector<T>::GeMM(const vector<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool vector<T>::GeMM(const matrix<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)
```

Fondamentalmente la funzione generale di moltiplicazione della matrice è descritta come:

```
C[m,n] = α *Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

con le seguenti dimensioni: matrice A è M x K, matrice B è K x N e matrice C è M x N.

Pertanto, le matrici dovrebbero essere compatibili per la moltiplicazione, cioè il numero di colonne nella prima matrice dovrebbe essere uguale al numero di righe nella seconda matrice. La moltiplicazione di matrice non è commutativa: il risultato della moltiplicazione della prima matrice per la seconda non è uguale al risultato della moltiplicazione della seconda matrice per la prima nel caso generale.

Esempio:

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//-- calcola GeMM per due vettori
   matrix_c.GeMM(vector_a, vector_b, 1, 0);
   Print("matrix_c:\n ", matrix_c, "\n");
   /*
   matrix_c:
    [[4,3,2,1]
    [8,6,4,2]
    [12,9,6,3]
    [16,12,8,4]
    [20,15,10,5]]
   */
//-- creare matrici come vettori
   matrix matrix_a(5, 1);
   matrix matrix_b(1, 4);
   matrix_a.Col(vector_a, 0);
   matrix_b.Row(vector_b, 0);
   Print("matrix_a:\n ", matrix_a);
   Print("matrix_b:\n ", matrix_b);
   /*
   matrix_a:
   [[1]
   [2]
   [3]
```

```
    [4]
    [5]]
    matrix_b:
    [[4,3,2,1]]
    */
//-- calcola GeMM per due matrici e ottieni lo stesso risultato
    matrix_c.GeMM(matrix_a, matrix_b, 1, 0);
    Print("matrix_c:\n ", matrix_c);
    /*
    matrix_c:
      [[4,3,2,1]
      [8,6,4,2]
      [12,9,6,3]
      [16,12,8,4]
      [20,15,10,5]]
    */
   }
```

Vedere anche

MatMul

Power

Eleva una matrice quadrata alla potenza intera.

```
matrix matrix::Power(
  const int  power      // potenza
   );
```

Parametri

power

[in] L'esponente può essere qualsiasi intero, positivo, negativo o zero.

Valore Restituito

Matrice.

Note

La matrice risultante ha le stesse dimensioni della matrice originale. L'elevamento alla potenza di 0, restituisce la matrice identità. La potenza positiva n significa che la matrice originale viene moltiplicata n volte per se stessa. La potenza negativa -n significa che la matrice originale viene prima invertita, e quindi la matrice invertita viene moltiplicata per se stessa n volte.

Un semplice algoritmo per elevare una matrice a una potenza in MQL5:

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//-- la matrice deve essere quadrata
   if(a.Rows()!=a.Cols())
      return(false);
//-- la dimensione della matrice risultante è esattamente la stessa
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//-- quando elevata a zero, restituisce la matrice identità
   if(power==0)
      result.Identity();
   else
     {
      //---per una misura negativa, prima invertire la matrice
      if(power<0)
        {
         matrix inverted=a.Inv();
         result=inverted;
         for(int i=-1; i>power; i--)
            result=result.MatMul(inverted);
        }
```

```
      else
        {
         result=a;
         for(int i=1; i<power; i++)
            result=result.MatMul(a);
        }
     }
//---
   c=result;
   return(true);
  }
```

Esempio MQL5:

```
  matrix i= {{0, 1}, {-1, 0}};
  Print("i:\n", i);

  Print("i.Power(3):\n", i.Power(3));

  Print("i.Power(0):\n", i.Power(0));

  Print("i.Power(-3):\n", i.Power(-3));

  /*
  i:
  [[0,1]
   [-1,0]]

  i.Power(3):
  [[0,-1]
   [1,0]]

  i.Power(0):
  [[1,0]
   [0,1]]

  i.Power(-3):
  [[0, -1]
   [1,0]]
  */
```

Esempio Python:

```
import numpy as np
from numpy.linalg import matrix_power

# matrix equiv. of the imaginary unit
```

```
i = np.array([[0, 1], [-1, 0]])
print("i:\n",i)

# should = -i
print("matrix_power(i, 3) :\n",matrix_power(i, 3) )

print("matrix_power(i, 0):\n",matrix_power(i, 0))

# should = 1/(-i) = i, but w/ f.p. elements
print("matrix_power(i, -3):\n",matrix_power(i, -3))
```

```
i:
 [[ 0  1]
 [-1  0]]

matrix_power(i, 3) :
 [[ 0 -1]
 [ 1  0]]

matrix_power(i, 0):
 [[1 0]
 [0 1]]

matrix_power(i, -3):
 [[ 0.  1.]
 [-1.  0.]]
```

Dot

Prodotto scalare di due vettori.

```
double vector::Dot(
  const vector&  b      // secondo vettore
   );
```

Parametri

b

[in] Vettore.

Valore Restituito

Scalare.

Note

Il prodotto scalare di 2 matrici e il prodotto della matrice matrix::MatMul().

Un semplice algoritmo per il prodotto scalare dei vettori in MQL5:

```
double VectorDot(const vector& vector_a, const vector& vector_b)
  {
   double dot=0;

   if(vector_a.Size()==vector_b.Size())
     {
      for(ulong i=0; i<vector_a.Size(); i++)
         dot+=vector_a[i]*vector_b[i];
     }

   return(dot);
  }
```

Esempio MQL5:

```
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
```

Esempio Python:

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

Kron

Restituisce il prodotto di Kronecker di due matrici, matrice e vettore, vettore e matrice o due vettori.

```
matrix matrix::Kron(
  const matrix&  b      // seconda matrice
   );

matrix matrix::Kron(
  const vector&  b      // vettore
   );

matrix vector::Kron(
  const matrix&  b      // matrice
   );

matrix vector::Kron(
  const vector&  b      // secondo vettore
   );
```

Parametri

b

[in] seconda matrice.

Valore Restituito

Matrice.

Note

Il prodotto di Kronecker è anche definito come moltiplicazione della matrice a blocchi.

Un semplice algoritmo per il prodotto di Kronecker per due matrici in MQL5:

```
matrix MatrixKronecker(const matrix& matrix_a,const matrix& matrix_b)
  {
   ulong  M=matrix_a.Rows();
   ulong  N=matrix_a.Cols();
   ulong  P=matrix_b.Rows();
   ulong  Q=matrix_b.Cols();
   matrix matrix_c(M*P,N*Q);

   for(ulong m=0; m<M; m++)
      for(ulong n=0; n<N; n++)
         for(ulong p=0; p<P; p++)
            for(ulong q=0; q<Q; q++)
```

```
                matrix_c[m*P+p][n*Q+q]=matrix_a[m][n] * matrix_b[p][q];

   return(matrix_c);
  }
```

Esempio MQL5:

```
   matrix a={{1,2,3},{4,5,6}};
   matrix b=matrix::Identity(2,2);
   vector v={1,2};

   Print(a.Kron(b));
   Print(a.Kron(v));

  /*
   [[1,0,2,0,3,0]
    [0,1,0,2,0,3]
    [4,0,5,0,6,0]
    [0,4,0,5,0,6]]

   [[1,2,2,4,3,6]
    [4,8,5,10,6,12]]
  */
```

Esempio Python:

```
import numpy as np

A = np.arange(1,7).reshape(2,3)
B = np.identity(2)
V = [1,2]
print(np.kron(A, B))
print("")
print(np.kron(A, V))

[[1. 0. 2. 0. 3. 0.]
 [0. 1. 0. 2. 0. 3.]
 [4. 0. 5. 0. 6. 0.]
 [0. 4. 0. 5. 0. 6.]]

[[ 1  2  2  4  3  6]
 [ 4  8  5 10  6 12]]
```

Inner

Prodotto interno di due matrici.

```
matrix matrix::Inner(
  const matrix&  b      // seconda matrice
   );
```

Parametri

b

[in] Matrice.

Valore Restituito

Matrice.

Note

Il prodotto interno per due vettori è il prodotto scalare dei due vettori vector::Dot().

Un semplice algoritmo per il prodotto interno di due matrici in MQL5:

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//-- il numero di colonne deve essere uguale
   if(a.Cols()!=b.Cols())
      return(false);
//-- la dimensione della matrice risultante dipende dal numero di vettori in ciascuna
   ulong  rows=a.Rows();
   ulong  cols=b.Rows();
   matrix result(rows,cols);
//---
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
//---
   c=result;
   return(true);
  }
```

Esempio MQL5:

```
   matrix a={{0,1,2},{3,4,5}};
   matrix b={{0,1,2},{3,4,5},{6,7,8}};
   matrix c=a.Inner(b);
   Print(c);
   matrix a1={{0,1,2}};
   matrix c1=a1.Inner(b);
   Print(c1);

  /*
  [[5,14,23]
  [14,50,86]]
  [[5,14,23]]
  */
```

Esempio Python:

```
import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))

import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))
```

Outer

Calcola il prodotto esterno di due matrici o due vettori.

```
matrix matrix::Outer(
  const matrix&  b      // seconda matrice
   );

matrix vector::Outer(
  const vector&  b      // secondo vettore
   );
```

Parametri

b

[in] Matrice.

Valore Restituito

Matrice.

Note

Il prodotto esterno, come il prodotto di Kronecker, è anche una moltiplicazione di matrice (e vettore) a blocchi.

Un semplice algoritmo per il prodotto esterno di due matrici in MQL5:

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//-- la dimensione della matrice risultante dipende dalle dimensioni della matrice
   ulong  rows=matrix_a.Rows()*matrix_a.Cols();
   ulong  cols=matrix_b.Rows()*matrix_b.Cols();
   matrix matrix_c(rows,cols);
   ulong  cols_a=matrix_a.Cols();
   ulong  cols_b=matrix_b.Cols();
//---
   for(ulong i=0; i<rows; i++)
     {
      ulong row_a=i/cols_a;
      ulong col_a=i%cols_a;
      for(ulong j=0; j<cols; j++)
        {
         ulong row_b=j/cols_b;
         ulong col_b=j%cols_b;
         matrix_c[i][j]=matrix_a[row_a][col_a] * matrix_b[row_b][col_b];
        }
```

```
      }
//---
   return(matrix_c);
  }
```

Esempio MQL5:

```
   vector vector_a={0,1,2,3,4,5};
   vector vector_b={0,1,2,3,4,5,6};
   Print("vector_a.Outer\n",vector_a.Outer(vector_b));
   Print("vector_a.Kron\n",vector_a.Kron(vector_b));

   matrix matrix_a={{0,1,2},{3,4,5}};
   matrix matrix_b={{0,1,2},{3,4,5},{6,7,8}};
   Print("matrix_a.Outer\n",matrix_a.Outer(matrix_b));
   Print("matrix_a.Kron\n",matrix_a.Kron(matrix_b));

  /*
   vector_a.Outer
   [[0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6]
    [0,2,4,6,8,10,12]
    [0,3,6,9,12,15,18]
    [0,4,8,12,16,20,24]
    [0,5,10,15,20,25,30]]
   vector_a.Kron
   [[0,0,0,0,0,0,0,0,1,2,3,4,5,6,0,2,4,6,8,10,12,0,3,6,9,12,15,18,0,4,8,12,16,20,24,0,
   matrix_a.Outer
   [[0,0,0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6,7,8]
    [0,2,4,6,8,10,12,14,16]
    [0,3,6,9,12,15,18,21,24]
    [0,4,8,12,16,20,24,28,32]
    [0,5,10,15,20,25,30,35,40]]
   matrix_a.Kron
   [[0,0,0,0,1,2,0,2,4]
    [0,0,0,3,4,5,6,8,10]
    [0,0,0,6,7,8,12,14,16]
    [0,3,6,0,4,8,0,5,10]
    [9,12,15,12,16,20,15,20,25]
    [18,21,24,24,28,32,30,35,40]]
   */
```

Esempio Python:

```
import numpy as np
```

```
A = np.arange(6)
B = np.arange(7)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
print(np.kron(A, B))

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
print("np.outer")
print(np.outer(A, B))
print("np.kron")

np.outer
[[ 0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6]
 [ 0  2  4  6  8 10 12]
 [ 0  3  6  9 12 15 18]
 [ 0  4  8 12 16 20 24]
 [ 0  5 10 15 20 25 30]]
np.kron
[ 0  0  0  0  0  0  0  0  1  2  3  4  5  6  0  2  4  6  8 10 12  0  3  6
  9 12 15 18  0  4  8 12 16 20 24  0  5 10 15 20 25 30]
np.outer
[[ 0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8]
 [ 0  2  4  6  8 10 12 14 16]
 [ 0  3  6  9 12 15 18 21 24]
 [ 0  4  8 12 16 20 24 28 32]
 [ 0  5 10 15 20 25 30 35 40]]
np.kron
[[ 0  0  0  0  1  2  0  2  4]
 [ 0  0  0  3  4  5  6  8 10]
 [ 0  0  0  6  7  8 12 14 16]
 [ 0  3  6  0  4  8  0  5 10]
 [ 9 12 15 12 16 20 15 20 25]
 [18 21 24 24 28 32 30 35 40]]
```

CorrCoef

Calcola il coefficiente di correlazione di Pearson (coefficiente di correlazione lineare).

```
matrix matrix::CorrCoef(
  const bool    rowvar=true  // righe o colonne del vettore in osservazione
);

scalar vector::CorrCoef(
  const vector&  b           // secondo vettore
);
```

Valore Restituito

Coefficiente di correlazione di Pearson.

Note

Il coefficiente di correlazione è nell'intervallo [-1, 1].

A causa dell'arrotondamento in virgola mobile, l'array risultante potrebbe non essere Hermitiano, gli elementi diagonali potrebbero non essere 1 e gli elementi potrebbero non soddisfare la disuguaglianza abs(a) <= 1. Le parti reali e immaginarie sono ritagliate all'intervallo [-1, 1] nel tentativo di migliorare quella situazione, ma non è di grande aiuto nel caso di complex.

Un semplice algoritmo per calcolare il coefficiente di correlazione di due vettori usando MQL5:

```
double VectorCorrelation(const vector& vector_x,const vector& vector_y)
  {
   ulong n=vector_x.Size()<vector_y.Size() ? vector_x.Size() : vector_y.Size();
   if(n<=1)
      return(0);

   ulong  i;
   double xmean=0;
   double ymean=0;
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_x[i]))
         return(0);
      if(!MathIsValidNumber(vector_y[i]))
         return(0);
      xmean+=vector_x[i];
      ymean+=vector_y[i];
     }
   xmean/=(double)n;
   ymean/=(double)n;
```

```
   double s=0;
   double xv=0;
   double yv=0;
   double t1=0;
   double t2=0;
//-- calcolo
   s=0;
   for(i=0; i<n; i++)
     {
      t1=vector_x[i]-xmean;
      t2=vector_y[i]-ymean;
      xv+=t1*t1;
      yv+=t2*t2;
      s+=t1*t2;
     }
//--- controllo
   if(xv==0 || yv==0)
      return(0);
//--- risultato restituito
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

Esempio MQL5:

```
   vectorf vector_a={1,2,3,4,5};
   vectorf vector_b={0,1,0.5,2,2.5};
   Print("vectors correlation ",vector_a.CorrCoef(vector_b));
//---
   matrixf matrix_a={{1,2,3,4,5},
                     {0,1,0.5,2,2.5}};
   Print("matrix rows correlation\n",matrix_a.CorrCoef());
   matrixf matrix_a2=matrix_a.Transpose();
   Print("transposed matrix cols correlation\n",matrix_a2.CorrCoef(false));
   matrixf matrix_a3={{1.0f, 2.0f, 3.0f, 4.0f, 5.0f},
                      {0.0f, 1.0f, 0.5f, 2.0f, 2.5f},
                      {0.1f, 1.0f, 2.0f, 1.0f, 0.3f}};
   Print("rows correlation\n",matrix_a3.CorrCoef());
   Print("cols correlation\n",matrix_a3.CorrCoef(false));

  /*
   vectors correlation 0.9149913787841797
   matrix rows correlation
   [[1,0.91499138]
    [0.91499138,1]]
   transposed matrix cols correlation
   [[1,0.91499138]
    [0.91499138,1]]
```

```
   rows correlation
   [[1,0.91499138,0.08474271]
    [0.91499138,1,-0.17123166]
    [0.08474271,-0.17123166,1]]
   cols correlation
   [[1,0.99587059,0.85375023,0.91129309,0.83773589]
    [0.99587059,1,0.80295509,0.94491106,0.88385159]
    [0.85375023,0.80295509,1,0.56362146,0.43088508]
    [0.91129309,0.94491106,0.56362146,1,0.98827404]
    [0.83773589,0.88385159,0.43088508,0.98827404,1]]
   */
```

Esempio Python:

```
import numpy as np
va=[1,2,3,4,5]
vb=[0,1,0.5,2,2.5]
print("vectors correlation")
print(np.corrcoef(va,vb))

ma=np.zeros((2,5))
ma[0,:]=va
ma[1,:]=vb
print("matrix rows correlation")
print(np.corrcoef(ma))
print("transposed matrix cols correlation")
print(np.corrcoef(np.transpose(ma),rowvar=False))
print("")

ma1=[[1,2,3,4,5],[0,1,0.5,2,2.5],[0.1,1,0.2,1,0.3]]
print("rows correlation\n",np.corrcoef(ma1))
print("cols correlation\n",np.corrcoef(ma1,rowvar=False))

transposed matrix cols correlation
[[1.         0.91499142]
 [0.91499142 1.        ]]

rows correlation
 [[1.         0.91499142 0.1424941 ]
 [0.91499142 1.         0.39657517]
 [0.1424941  0.39657517 1.        ]]
cols correlation
 [[1.         0.99587059 0.98226063 0.91129318 0.83773586]
 [0.99587059 1.         0.99522839 0.94491118 0.88385151]
 [0.98226063 0.99522839 1.         0.97234063 0.92527551]
 [0.91129318 0.94491118 0.97234063 1.         0.98827406]
 [0.83773586 0.88385151 0.92527551 0.98827406 1.        ]]
```

Cov

Calcola la matrice di covarianza.

```
matrix matrix::Cov(
  const bool    rowvar=true  // righe o colonne del vettore in osservazione
);

matrix vector::Cov(
  const vector&  b           // secondo vettore
);
```

Parametri

b

[in] Secondo vettore.

Note

Calcola la matrice di covarianza.

Un semplice algoritmo per calcolare la matrice di covarianza di due vettori usando MQL5:

```
bool VectorCovariation(const vector& vector_a,const vector& vector_b,matrix& matrix_c)
  {
   int i,j;
   int m=2;
   int n=(int)(vector_a.Size()<vector_b.Size()?vector_a.Size():vector_b.Size());
//--- controlli
   if(n<=1)
      return(false);
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_a[i]))
         return(false);
      if(!MathIsValidNumber(vector_b[i]))
         return(false);
     }
//---
   matrix matrix_x(2,n);
   matrix_x.Row(vector_a,0);
   matrix_x.Row(vector_b,1);
   vector t=vector::Zeros(m);
//-- calcolo
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         t[i]+=matrix_x[i][j]/double(n);
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
```

```
         matrix_x[i][j]-=t[i];
//--- syrk C=alpha*A^H*A+beta*C (beta=0 e non considerato)
   matrix_c=matrix::Zeros(m,m);
   for(i=0; i<m; i++)
     {
      for(j=0; j<n; j++)
        {
         double v=matrix_x[i][j]/(n-1);
         for(int i_=i; i_<m; i_++)
            matrix_c[i][i_]+=v*matrix_x[i_][j];
        }
     }
//-- simmetria forzata
   for(i=0; i<m-1; i++)
      for(j=i+1; j<m; j++)
         matrix_c[j][i]=matrix_c[i][j];
//---
   return(true);
  }
```

Esempio MQL5:

```
   matrix matrix_a={{3,-2.1},{1.1,-1},{0.12,4.3}};
   Print("covariation cols\n",matrix_a.Cov(false));
   Print("covariation rows\n",matrix_a.Cov());

   vector vector_a=matrix_a.Col(0);
   vector vector_b=matrix_a.Col(1);
   Print("covariation vectors\n",vector_a.Cov(vector_b));

  /*
   covariation cols
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
   covariation rows
   [[13.005,5.355,-10.659]
    [5.355,2.205,-4.389]
    [-10.659,-4.389,8.736199999999998]]
   covariation vectors
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
  */
```

Esempio Python:

```
import numpy as np
matrix_a=np.array([[3,-2.1],[1.1,-1],[0.12,4.3]])
```

```
matrix_c=np.cov(matrix_a,rowvar=False)
print("covariation cols\n",matrix_c)
matrix_c2=np.cov(matrix_a)
print("covariation rows\n",matrix_c2)

vector_a=matrix_a[:,0]
vector_b=matrix_a[:,1]
matrix_c3=np.cov(vector_a,vector_b)
print("covariation vectors\n",matrix_c3)

covariation cols
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
covariation rows
 [[ 13.005    5.355  -10.659 ]
 [  5.355    2.205   -4.389 ]
 [-10.659   -4.389    8.7362]]
covariation vectors
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
```

Correlate

Calcola la correlazione incrociata di due vettori.

```
vector vector::Correlate(
  const vector&            v,        // vettore
  ENUM_VECTOR_CONVOLVE     mode      // modalità
   );
```

Parametri

v

[in] Secondo vettore.

mode

[in] Il parametro 'mode' determina la modalità di calcolo della convoluzione lineare. Valore dall'enumerazione ENUM_VECTOR_CONVOLVE.

Valore Restituito

Correlazione incrociata di due vettori.

Note

Il parametro 'mode' determina la modalità di calcolo della convoluzione lineare.

Un semplice algoritmo per calcolare il coefficiente di correlazione di due vettori usando MQL5:

```
vector VectorCrossCorrelationFull(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[n-i-1]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationSame(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
   vector c=vector::Zeros(size);
```

```
   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationValid(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
```

Esempio MQL5:

```
   vector a={1,2,3,4,5};
   vector b={0,1,0.5};

   Print("full\n",a.Correlate(b,VECTOR_CONVOLVE_FULL));
   Print("same\n",a.Correlate(b,VECTOR_CONVOLVE_SAME));
   Print("valid\n",a.Correlate(b,VECTOR_CONVOLVE_VALID));
   Print("full\n",b.Correlate(a,VECTOR_CONVOLVE_FULL));

  /*
   full
```

```
   [0.5,2,3.5,5,6.5,5,0]
   same
   [2,3.5,5,6.5,5]
   valid
   [3.5,5,6.5]
   full
   [0,5,6.5,5,3.5,2,0.5]
  */
```

Esempio Python:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.correlate(a,b,'full'))
print("same\n",np.correlate(a,b,'same'));
print("valid\n",np.correlate(a,b,'valid'));
print("full\n",np.correlate(b,a,'full'))

full
 [0.5 2.  3.5 5.  6.5 5.  0. ]
same
 [2.  3.5 5.  6.5 5. ]
valid
 [3.5 5.  6.5]
full
 [0.  5.  6.5 5.  3.5 2.  0.5]
```

Convolve

Restituisce la convoluzione lineare discreta di due vettori

```
vector vector::Convolve(
  const vector&            v,         // vettore
  ENUM_VECTOR_CONVOLVE     mode       // modalità
   );
```

Parametri

v

[out] Secondo vettore.

mode

[in] Il parametro 'mode' determina la modalità di calcolo della convoluzione lineare ENUM_VECTOR_CONVOLVE.

Valore Restituito

Convoluzione lineare discreta di due vettori.

Un semplice algoritmo per calcolare la convoluzione di due vettori in MQL5:

```
vector VectorConvolutionFull(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionFull(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[i]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionSame(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionSame(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
```

```
   int    size=MathMax(m,n);
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionValid(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionValid(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
```

Esempio MQL5:

```
  vector a= {1, 2, 3, 4, 5};
  vector b= {0, 1, 0.5};

  Print("full\n", a.Convolve(b, VECTOR_CONVOLVE_FULL));
  Print("same\n", a.Convolve(b, VECTOR_CONVOLVE_SAME));
```

```
  Print("valid\n", a.Convolve(b, VECTOR_CONVOLVE_VALID));

  /*
   full
   [0,1,2.5,4,5.5,7,2.5]
   same
   [1,2.5,4,5.5,7]
   valid
   [2.5,4,5.5]
  */
```

Esempio Python:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.convolve(a,b,'full'))
print("same\n",np.convolve(a,b,'same'));
print("valid\n",np.convolve(a,b,'valid'));


full
 [0.  1.  2.5 4.  5.5 7.  2.5]
same
 [1.  2.5 4.  5.5 7. ]
valid
 [2.5 4.  5.5]
```

Trasformazioni di matrici

La decomposizione della matrice può essere utilizzata nei seguenti casi:

- come passo intermedio nella risoluzione di sistemi di equazioni lineari
- per l'inversa della matrice
- nel calcolo dei determinanti
- quando cerchiamo gli autovalori e autovettori di una matrice
- quando si calcolano le funzioni analitiche delle matrici
- quando si utilizza il metodo dei minimi quadrati
- nella soluzione numerica delle equazioni differenziali

Differenti tipi di decomposizione della matrice vengono utilizzati in base al problema.

Funzione	Azione
Cholesky	Calcola la decomposizione di Cholesky
Eig	Calcola gli autovalori e gli autovettori di destra di una matrice quadrata
EigVals	Calcola gli autovalori di una matrice generica
LU	Fattorizzazione LU di una matrice come prodotto di una matrice triangolare inferiore e di una matrice triangolare superiore
LUP	Fattorizzazione LU con pivoting parziale, che si riferisce alla decomposizione LU con permutazioni di riga solo: PA=LU
QR	Calcola la fattorizzazione qr di una matrice
SVD	Decomposizione ai Valori Singolari

Cholesky

Calcola la decomposizione di Cholesky.

```
bool matrix::Cholesky(
  matrix&  L      // matrice
   );
```

Parametri

L key

[out] Matrice triangolare inferiore.

Valore di Ritorno

Restituisce true in caso di successo, altrimenti false.

Note

Restituisce la decomposizione Cholesky, L * L.H, della matrice quadrata a, dove L è triangolare inferiore e .H è l'operatore di trasposizione coniugata (che è la trasposta ordinaria se a ha valori reali). a deve essere Hermitiano (simmetrico se ha valori reali) e definito positivo. Non viene eseguito alcun controllo per verificare se a è Hermitiano o meno. Inoltre, vengono utilizzati solo gli elementi triangolari inferiori e diagonali di a . Viene effettivamente restituito solo L .

Esempio

```
matrix matrix_a= {{5.7998084, -2.1825367}, {-2.1825367, 9.85910595}};
matrix matrix_l;
Print("matrix_a\n", matrix_a);

matrix_a.Cholesky(matrix_l);
Print("matrix_l\n", matrix_l);
Print("check\n", matrix_l.MatMul(matrix_l.Transpose()));

/*
matrix_a
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
matrix_l
[[2.408279136645086,0]
 [-0.9062640068544704,3.006291985133859]]
check
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
*/
```

Eig

Calcola gli autovalori e gli autovettori di destra di una matrice quadrata.

```
bool matrix::Eig(
  matrix&  eigen_vectors,     // matrice di autovettori
  vector&  eigen_values       // vettore di autovalori
   );
```

Parametri

eigen_vectors

[out] Matrice di autovettori verticali.

eigen_values

[out] Vettore di autovalori.

Valore di Ritorno

Restituisce true in caso di successo, altrimenti false.

Esempio

```
#property script_show_inputs
//--- parametri di input
input int  InpSize1   =512;
input int  InpSize2   =256;
input int  InpSize3   =1024;
//+------------------------------------------------------------------+
//| Riempimento della matrice invertibile quadrata di prova          |
//+------------------------------------------------------------------+
template<typename T>
void MatrixFill(matrix<T> &matrix_a)
 {
  ulong size_m=matrix_a.Rows();
  ulong size_k=matrix_a.Cols();
  T     value=0.0;
//--- riempire la matrice
  for(ulong i=0; i<size_m; i++)
   {
    for(ulong j=0; j<size_k; j++)
     {
      if(i==j)
        matrix_a[i][j]=T(1.0+i);
      else
       {
        value+=1.0;
        matrix_a[i][j]=value;
       }
     }
```

```
   }
 }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
 {
  int errors=0;

  errors+=TestEigen<double>(InpSize1);
  errors+=TestEigen<double>(InpSize2);
  errors+=TestEigen<double>(InpSize3);

  errors+=TestEigen<float>(InpSize1);
  errors+=TestEigen<float>(InpSize2);
  errors+=TestEigen<float>(InpSize3);
//---
  Print("Test ", errors?"failed":"passed");
  return(errors);
 }

/*
Risultato

Eigen solver of double matrix 512 x 512  passed  vectors=489  time=4268.506 ms
Eigen solver of double matrix 256 x 256  passed  vectors=251  time=417.610 ms
Eigen solver of double matrix 1024 x 1024  passed  vectors=916  time=43708.280 ms
Eigen solver of float matrix 512 x 512  passed  vectors=1  time=2508.357 ms
Eigen solver of float matrix 256 x 256  passed  vectors=1  time=188.859 ms
Eigen solver of float matrix 1024 x 1024  passed  vectors=1  time=27209.666 ms
Test passed
*/

//+------------------------------------------------------------------+
//| Verifica del metodo Eig                                          |
//+------------------------------------------------------------------+
template<typename T>
int TestEigen(const int size_m)
 {
  int vectors=0;
  matrix<T> matrix_a(size_m, size_m);
  matrix<T> matrix_v(size_m, size_m);
  vector<T> vector_e(size_m);
//--- popolare la matrice quadrata
  MatrixFill(matrix_a);
//--- microsecondi che saranno misurati
  ulong t1=GetMicrosecondCount();
//--- Autorisolutore
  matrix_a.Eig(matrix_v, vector_e);
```

```
//--- misura
  ulong t2=GetMicrosecondCount();
//--- controlla se A * v = lambda * v
  for(ulong n=0; n<vector_e.Size(); n++)
   {
    vector<T> eigen_vector=matrix_v.Col(n);
    vector<T> vector_c1   =eigen_vector*vector_e[n];
    vector<T> vector_c2   =matrix_a.MatMul(eigen_vector);

    //--- troppe divisioni, indebolire il controllo di precisione fino alla decima ci
    ulong errors=vector_c1.CompareByDigits(vector_c2, sizeof(T)==sizeof(double) ? 10 :
    if(int(errors)<size_m/10)
      vectors++;
   }
  double elapsed_time=double(t2-t1)/1000.0;
  printf("Eigen solver of %s matrix %d x %d  %s  vectors=%d  time=%.3f ms",
         typename(T), size_m, size_m, vectors>0?"passed":"failed", vectors, elapsed_t
  return(vectors==0);
 }
```

EigVals

Calcola gli autovalori di una matrice generica

```
bool matrix::EigVals(
  vector&  eigen_values      // vettore di autovalori
   );
```

Parametri

eigen_values

[out] Vettore degli autovalori destri.

Valore di Ritorno

Restituisce true in caso di successo, altrimenti false.

Note

L'unica differenza tra EigVals ed Eig è che EigVals non calcola gli autovettori, ma calcola solo gli autovalori.

LU

La fattorizzazione LU di una matrice come prodotto di una matrice triangolare inferiore e di una matrice triangolare superiore.

```
bool matrix::LU(
  matrix&  L,      // matrice triangolare inferiore
  matrix&  U       // matrice triangolare superiore
   );
```

Parametri

L key

[out] Matrice triangolare inferiore.

U

[out] Matrice triangolare superiore.

Valore di Ritorno

Restituisce true in caso di successo, altrimenti false.

Esempio

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- decomposizione LU
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- controlla se A = L * U
   Print("check\n",matrix_l.MatMul(matrix_u));


   /*
   matrix_l
   [[1,0,0,0]
    [5,1,0,0]
    [8,0.875,1,0]
    [11,1.25,0.5904761904761905,1]]
   matrix_u
   [[1,2,3,4]
    [0,-8,-9,-13]
    [0,0,-13.125,-10.625]
    [0,0,0,-17.47619047619047]]
   check
   [[1,2,3,4]
```

```
    [5,2,6,7]
    [8,9,3,10]
    [11,12,14,4]]
   */
```

LUP

Fattorizzazione LU con pivoting parziale, che si riferisce alla decomposizione LU con permutazioni di riga solo: PA=LU

```
bool  LUP(
  matrix&  L,      // matrice triangolare inferiore
  matrix&  U,       // matrice triangolare superiore
  matrix&  P       // matrice di permutazione
   );
```

Parametri

L key

[out] Matrice triangolare inferiore.

U

[out] Matrice triangolare superiore.

P

[out] Matrice di permutazione

Valore di Ritorno

Restituisce true in caso di successo, altrimenti false.

Esempio

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u,matrix_p;
//--- decomposizione LUP
   matrix_a.LUP(matrix_l,matrix_u,matrix_p);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
   Print("matrix_p\n",matrix_p);
//--- controlla se P * A = L * U
   Print("P * A\n",matrix_p.MatMul(matrix_a));
   Print("L * U\n",matrix_l.MatMul(matrix_u));

   /*
   matrix_l
   [[1,0,0,0]
    [0.4545454545454545,1,0,0]
    [0.7272727272727273,-0.07894736842105282,1,0]
    [0.09090909090909091,-0.2631578947368421,-0.2262773722627738,1]]
   matrix_u
   [[11,12,14,4]
```

```
   [0,-3.454545454545454,-0.3636363636363633,5.181818181818182]
   [0,0,-7.210526315789473,7.500000000000001]
   [0,0,0,6.697080291970803]]
  matrix_p
  [[0,0,0,1]
   [0,1,0,0]
   [0,0,1,0]
   [1,0,0,0]]
  P * A
  [[11,12,14,4]
   [5,2,6,7]
   [8,9,3,10]
   [1,2,3,4]]
  L * U
  [[11,12,14,4]
   [5,2,6,7]
   [8,9,3.000000000000001,10]
   [1,2,3,4]]
  */
```

QR

Calcola la fattorizzazione qr di una matrice

```
bool  QR(
  matrix&  Q,      // matrice con colonne ortonormali
  matrix&  R       // matrice triangolare superiore
   );
```

Parametri

Q

[out] Una matrice con colonne ortonormali. Quando mode = 'complete' il risultato è una matrice ortogonale/unitaria a seconda che a sia o meno reale/complesso. Il determinante può essere +/- 1 in quel caso. Nel caso in cui il numero di dimensioni nell'array di input è maggiore di 2, viene restituita una pila di matrici con le proprietà sopra riportate.

R key

[out] Matrice triangolare superiore.

Valore di Ritorno

Restituisce true in caso di successo, altrimenti false.

Esempio

```
//---  A*x = b
  matrix A = {{0, 1}, {1, 1}, {1, 1}, {2, 1}};
  Print("A \n", A);
  vector b = {1, 2, 2, 3};
  Print("b \n", b);
//--- A = Q*R
  matrix q, r;
  A.QR(q, r);
  Print("q \n", q);
  Print("r \n", r);
  matrix qr=q.MatMul(r);
  Print("qr \n", qr);
  /*
  A
  [[0,1]
  [1,1]
  [1,1]
  [2,1]]
  b
  [1,2,2,3]
  q
  [[0.4082482904638631,-0.8164965809277259,-1.110223024625157e-16,-0.4082482904638631]
  [0.4625425214347352,-0.0374574785526496,0.7041241452319315,0.5374574785652647]
  [-0.5374574785652648,-0.0374574785526496,0.7041241452319316,-0.4625425214347352]
```

```
[-0.5749149571305296,-0.5749149571305299,-0.09175170953613698,0.5749149571305296]]
r
[[-1.224744871391589,-0.2415816237971962]
[-1.22474487139159,-1.466326495188786]
[1.224744871391589,1.316496580927726]
[1.224744871391589,0.2415816237971961]]
qr
[[-1.110223024625157e-16,1]
[1,0.9999999999999999]
[1,1]
[2,1]]
*/
```

SVD

Decomposizione ai Valori Singolari.

```
bool matrix::SVD(
  matrix&  U,                    // matrice unitaria
  matrix&  V,                    // matrice unitaria
  vector&  singular_values       // vettore dei valori singolari
   );
```

Parametri

U

[out] Matrice unitaria di ordine m, costituita da vettori singolari sinistri.

V

[out] Matrice unitaria di ordine n, costituita da vettori singolari destri.

singular_values

[out] Valori singolari

Valore di Ritorno

Restituisce true in caso di successo, altrimenti false.

Esempio

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
//---esecuzione decomposizione SVD
  matrix U, V;
  vector singular_values;
  b.SVD(U, V, singular_values);
  Print("U \n", U);
  Print("V \n", V);
  Print("singular_values = ", singular_values);

// blocco di controllo
//--- U * diagonale singolare * V = A
  matrix matrix_s;
  matrix_s.Diag(singular_values);
  Print("matrix_s \n", matrix_s);
  matrix matrix_vt=V.Transpose();
  Print("matrix_vt \n", matrix_vt);
  matrix matrix_usvt=(U.MatMul(matrix_s)).MatMul(matrix_vt);
  Print("matrix_usvt \n", matrix_usvt);
```

```
  ulong errors=(int)b.Compare(matrix_usvt, 1e-9);
  double res=(errors==0);
  Print("errors=", errors);

//--- un'altro controllo
  matrix U_Ut=U.MatMul(U.Transpose());
  Print("U_Ut \n", U_Ut);
  Print("Ut_U \n", (U.Transpose()).MatMul(U));

  matrix vt_V=matrix_vt.MatMul(V);
  Print("vt_V \n", vt_V);
  Print("V_vt \n", V.MatMul(matrix_vt));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
   [-1,0,1]
   [2,3,4]]
  U
  [[-0.7071067811865474,0.5773502691896254,0.408248290463863]
   [-6.827109697437648e-17,0.5773502691896253,-0.8164965809277256]
   [0.7071067811865472,0.5773502691896255,0.4082482904638627]]
  V
  [[0.5773502691896258,-0.7071067811865474,-0.408248290463863]
   [0.5773502691896258,1.779939029415334e-16,0.8164965809277258]
   [0.5773502691896256,0.7071067811865474,-0.408248290463863]]
  singular_values = [7.348469228349533,2.449489742783175,3.277709923350408e-17]

  matrix_s
  [[7.348469228349533,0,0]
   [0,2.449489742783175,0]
   [0,0,3.277709923350408e-17]]
  matrix_vt
  [[0.5773502691896258,0.5773502691896258,0.5773502691896256]
   [-0.7071067811865474,1.779939029415334e-16,0.7071067811865474]
   [-0.408248290463863,0.8164965809277258,-0.408248290463863]]
  matrix_usvt
  [[-3.999999999999997,-2.999999999999999,-2]
   [-0.9999999999999981,-5.977974170712231e-17,0.9999999999999974]
   [2,2.999999999999999,3.999999999999996]]
  errors=0

  U_Ut
  [[0.9999999999999993,-1.665334536937735e-16,-1.665334536937735e-16]
   [-1.665334536937735e-16,0.9999999999999987,-5.551115123125783e-17]
   [-1.665334536937735e-16,-5.551115123125783e-17,0.9999999999999999]]
```

```
  Ut_U
  [[0.9999999999999993,-5.551115123125783e-17,-1.110223024625157e-16]
   [-5.551115123125783e-17,0.9999999999999987,2.498001805406602e-16]
   [-1.110223024625157e-16,2.498001805406602e-16,0.999999999999999]]
  vt_V
  [[1,-5.551115123125783e-17,0]
   [-5.551115123125783e-17,0.9999999999999996,1.110223024625157e-16]
   [0,1.110223024625157e-16,0.9999999999999996]]
  V_vt
  [[0.9999999999999999,1.110223024625157e-16,1.942890293094024e-16]
   [1.110223024625157e-16,0.9999999999999998,1.665334536937735e-16]
   [1.942890293094024e-16,1.665334536937735e-16,0.9999999999999996]
  */
 }
```

Metodi statistici

Metodi di calcolo delle statistiche descrittive di matrici e vettori.

Funzione	Azione
ArgMax	Restituisce l'indice del valore massimo
ArgMin	Restituisce l'indice del valore minimo
Max	Restituisce il valore massimo in una matrice/vettore
Min	Restituisce il valore minimo in una matrice/vettore
Ptp	Restituisce l'intervallo di valori di una matrice/vettore o dell'asse della matrice data
Sum	Restituisce la somma degli elementi della matrice/vettore che possono essere eseguiti anche per l'asse dato (assi)
Prod	Restituisce il prodotto degli elementi della matrice/vettore che possono essere eseguiti anche per l'asse dato
CumSum	Restituisce la somma cumulativa degli elementi della matrice/vettore, compresi quelli lungo l'asse indicato
CumProd	Restituisce il prodotto cumulativo degli elementi della matrice/vettore, compresi quelli lungo l'asse indicato
Percentile	Restituisce il percentile specificato dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse specificato
Quantile	Restituisce il quantile specificato dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse specificato
Median	Calcola la mediana degli elementi matrice/vettore
Mean	Calcola la media aritmetica dei valori degli elementi
Average	Calcola la media ponderata dei valori della matrice/vettore
Std	Restituisce la deviazione standard dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse dato
Var	Calcola la varianza dei valori degli elementi della matrice/vettore
LinearRegression	Calcola un vettore/matrice con i valori di regressione lineare calcolati

ArgMax

Restituisce l'indice del valore massimo.

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Indice del valore massimo.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.ArgMax(0);
   vector rows_max=matrix_a.ArgMax(1);
   ulong  matrix_max=matrix_a.ArgMax();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max index ",matrix_max,"  max value ",matrix_a.Flat(matrix_max));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[0,3,1]
   rows_max=[0,2,0,1]
   max index 5  max value 12.0
   */
```

ArgMin

Restituisce l'indice del valore minimo.

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Indice del valore minimo.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.ArgMin(0);
   vector rows_min=matrix_a.ArgMin(1);
   ulong  matrix_min=matrix_a.ArgMin();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min index ",matrix_min,"  min value ",matrix_a.Flat(matrix_min));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,0,0]
   rows_min=[2,0,2,0]
   min index 3  min value 1.0
   */
```

Max

Restituisce il valore massimo in una matrice/vettore.

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Valore massimo in una matrice/vettore.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.Max(0);
   vector rows_max=matrix_a.Max(1);
   double matrix_max=matrix_a.Max();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max value ",matrix_max);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[10,11,12]
   rows_max=[10,12,6,11]
   max value 12.0
   */
```

Min

Restituisce il valore minimo in una matrice/vettore.

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Il valore minimo in una matrice/vettore.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.Min(0);
   vector rows_min=matrix_a.Min(1);
   double matrix_min=matrix_a.Min();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min value ",matrix_min);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,3,2]
   rows_min=[2,1,4,7]
   min value 1.0
   */
```

Ptp

Restituisce l'intervallo di valori di una matrice/vettore o dell'asse della matrice data, equivalente a Max() - Min(). Ptp - Picco a picco.

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Vettore con l'intervallo dei valori (massimo - minimo).

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_ptp=matrix_a.Ptp(0);
   vector rows_ptp=matrix_a.Ptp(1);
   double matrix_ptp=matrix_a.Ptp();

   Print("cols_ptp  ",cols_ptp);
   Print("rows_ptp  ",rows_ptp);
   Print("ptp value  ",matrix_ptp);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_ptp [9,8,10]
   rows_ptp [8,11,2,4]
   ptp value  11.0
   */
```

Sum

Restituisce la somma degli elementi della matrice/vettore che possono essere eseguiti anche per l'asse dato (assi).

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Somma degli elementi della matrice/vettore che possono essere eseguiti anche per l'asse dato (assi).

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_sum=matrix_a.Sum(0);
   vector rows_sum=matrix_a.Sum(1);
   double matrix_sum=matrix_a.Sum();

   Print("cols_sum=",cols_sum);
   Print("rows_sum=",rows_sum);
   Print("sum value ",matrix_sum);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_sum=[24,27,27]
   rows_sum=[15,21,15,27]
   sum value 78.0
   */
```

Prod

Restituisce il prodotto degli elementi della matrice/vettore che possono essere eseguiti anche per l'asse dato.

```
double vector::Prod(
  const double  initial=1      // moltiplicatore iniziale
   );

double matrix::Prod(
  const double  initial=1      // moltiplicatore iniziale
   );

vector matrix::Prod(
  const int     axis,          // asse
  const double  initial=1      // moltiplicatore iniziale
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

initial=1

[in] Moltiplicatore iniziale.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_prod=matrix_a.Prod(0);
   vector rows_prod=matrix_a.Prod(1);
   double matrix_prod=matrix_a.Prod();

   Print("cols_prod=",cols_prod);
   cols_prod=matrix_a.Prod(0,0.1);
   Print("cols_prod=",cols_prod);
   Print("rows_prod=",rows_prod);
   Print("prod value ",matrix_prod);

  /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_prod=[420,1320,864]
   cols_prod=[42,132,86.400000000000001]
```

```
   rows_prod=[60,96,120,693]
   prod value 479001600.0
  */
```

CumSum

Restituisce la somma cumulativa degli elementi della matrice/vettore, quelli compresi lungo l'asse indicato.

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Somma cumulativa degli elementi lungo l'asse indicato.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumsum=matrix_a.CumSum(0);
   matrix rows_cumsum=matrix_a.CumSum(1);
   vector cumsum_values=matrix_a.CumSum();

   Print("cols_cumsum\n",cols_cumsum);
   Print("rows_cumsum\n",rows_cumsum);
   Print("cumsum values  ",cumsum_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumsum
   [[10,3,2]
    [11,11,14]
    [17,16,18]
    [24,27,27]]
   rows_cumsum
   [[10,13,15]
    [1,9,21]
```

```
    [6,11,15]
    [7,18,27]]
   cumsum values  [10,13,15,16,24,36,42,47,51,58,69,78]
   */
```

CumProd

Restituisce il prodotto cumulativo degli elementi della matrice/vettore, compresi quelli lungo l'asse indicato.

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale per ogni colonna (cioè, sopra le righe), 1 — asse verticale per ogni riga (cioè sopra le colonne)

Valore Restituito

Prodotto cumulativo degli elementi lungo l'asse indicato.

Esempio

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumprod=matrix_a.CumProd(0);
   matrix rows_cumprod=matrix_a.CumProd(1);
   vector cumprod_values=matrix_a.CumProd();

   Print("cols_cumprod\n",cols_cumprod);
   Print("rows_cumprod\n",rows_cumprod);
   Print("cumprod values  ",cumprod_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumprod
   [[10,3,2]
    [10,24,24]
    [60,120,96]
    [420,1320,864]]
   rows_cumprod
   [[10,30,60]
```

```
    [1,8,96]
    [6,30,120]
    [7,77,693]]
   cumprod values  [10,30,60,60,480,5760,34560,172800,691200,4838400,53222400,47900160
   */
```

Percentile

Restituisce il percentile specificato dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse specificato.

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // asse
   );
```

Parametri

percent

[in] Percentili da calcolare, che devono essere compresi tra 0 e 100.

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Percentile: scalare o vettoriale.

Note

I valori validi del parametro 'percent' sono nell'intervallo [0, 100]. Viene utilizzato un algoritmo lineare per calcolare i percentili. Il calcolo corretto dei percentili richiede che la sequenza sia ordinata.

Esempio

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_percentile=matrix_a.Percentile(50,0);
   vectorf rows_percentile=matrix_a.Percentile(50,1);
   float matrix_percentile=matrix_a.Percentile(50);

   Print("cols_percentile ",cols_percentile);
   Print("rows_percentile ",rows_percentile);
   Print("percentile value  ",matrix_percentile);
```

```
/*
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_percentile [5.5,6.5,7.5]
rows_percentile [2,5,8,11]
percentile value  6.5
*/
```

Quantile

Restituisce il quantile specificato dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse specificato.

```
double vector::Quantile(
  const double quantile      // quantile
   );

double matrix::Quantile(
  const double quantile      // quantile
   );

vector matrix::Quantile(
  const double quantile,     // quantile
  const int    axis          // asse
   );
```

Parametri

quantile

[in] Quantile da calcolare, che deve essere compreso tra 0 e 1.

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Quantile: scalare o vettoriale.

Note

Il parametro 'quantile' prende valori nell'intervallo [0, 1]. Viene utilizzato un algoritmo lineare per calcolare i quantili. Il calcolo corretto dei quantili richiede che la sequenza sia ordinata.

Esempio

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_quantile=matrix_a.Quantile(0.5,0);
   vectorf rows_quantile=matrix_a.Quantile(0.5,1);
   float matrix_quantile=matrix_a.Quantile(0.5);

   Print("cols_quantile ",cols_quantile);
   Print("rows_quantile ",rows_quantile);
   Print("quantile value  ",matrix_quantile);

   /*
   matrix_a
```

```
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   cols_quantile [5.5,6.5,7.5]
   rows_quantile [2,5,8,11]
   quantile value  6.5
   */
```

Median

Calcola la mediana degli elementi della matrice/vettore.

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Mediana: scalare o vettoriale.

Note

La mediana è il valore centrale che separa la metà più alta degli elementi di matrice/vettore dalla metà più bassa degli elementi. Come Quantile(0.5) e Percentile(50). Il calcolo corretto della mediana richiede che la sequenza sia ordinata.

Esempio

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_median=matrix_a.Median(0);
   vectorf rows_median=matrix_a.Median(1);
   float matrix_median=matrix_a.Median();

   Print("cols_median ",cols_median);
   Print("rows_median ",rows_median);
   Print("median value  ",matrix_median);


   /*
   matrix_a
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   cols_median [5.5,6.5,7.5]
```

```
   rows_median [2,5,8,11]
   median value  6.5
   */
```

Mean

Calcola la media aritmetica dei valori degli elementi.

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Media aritmetica dei valori degli elementi.

Esempio

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_mean=matrix_a.Mean(0);
   vectorf rows_mean=matrix_a.Mean(1);
   float matrix_mean=matrix_a.Mean();

   Print("cols_mean ",cols_mean);
   Print("rows_mean ",rows_mean);
   Print("mean value  ",matrix_mean);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_mean [6,6.75,6.75]
   rows_mean [5,7,5,9]
   mean value  6.5
   */
```

Average

Calcola la media ponderata dei valori della matrice/vettore.

```
double vector::Average(
  const vector& weigts      // vettore ponderato
   );

double matrix::Average(
  const matrix& weigts      // matrice ponderata
);

vector matrix::Average(
  const matrix& weigts,     // matrice ponderata
  const int     axis        // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Media aritmetica: scalare o vettoriale.

Note

La matrice/vettore ponderata è associata alla matrice/vettore principale.

Esempio

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   matrixf matrix_w=matrixf::Ones(4,3);
   Print("matrix_a\n",matrix_a);

   vectorf cols_average=matrix_a.Average(matrix_w,0);
   vectorf rows_average=matrix_a.Average(matrix_w,1);
   float matrix_average=matrix_a.Average(matrix_w);

   Print("cols_average ",cols_average);
   Print("rows_average ",rows_average);
   Print("average value  ",matrix_average);


   /*
   matrix_a
   [[10,3,2]
```

```
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_average [6,6.75,6.75]
   rows_average [5,7,5,9]
   average value  6.5
   */ value  6.5
```

Std

Restituisce la deviazione standard dei valori degli elementi della matrice/vettore o degli elementi lungo l'asse dato.

```
double vector::Std();

double matrix::Std();

vector matrix::Std(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Deviazione standard: scalare o vettoriale.

Note

La deviazione standard è la radice quadrata della media delle deviazioni quadrate dalla media, cioè, std = sqrt(mean(x)), dove x = abs(a - a.mean())**2.

La deviazione quadrata media è tipicamente calcolata come x.sum() / N, dove N = len(x).

Esempio

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_std=matrix_a.Std(0);
   vectorf rows_std=matrix_a.Std(1);
   float matrix_std=matrix_a.Std();

   Print("cols_std ",cols_std);
   Print("rows_std ",rows_std);
   Print("std value  ",matrix_std);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
```

```
   cols_std [3.2403703,3.0310888,3.9607449]
   rows_std [3.5590262,4.5460606,0.81649661,1.6329932]
   std value  3.452052593231201
   */
```

Var

Calcola la varianza dei valori degli elementi della matrice/vettore.

```
double vector::Var();

double matrix::Var();

vector matrix::Var(
  const int  axis      // asse
   );
```

Parametri

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

Varianza: scalare o vettoriale.

Note

La varianza è la media delle deviazioni al quadrato dalla media, cioè, var = mean(x), dove x = abs(a - a.mean())**2.

La media è tipicamente calcolata come x.sum() / N, dove N = len(x).

Esempio

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_var=matrix_a.Var(0);
   vectorf rows_var=matrix_a.Var(1);
   float matrix_var=matrix_a.Var();

   Print("cols_var ",cols_var);
   Print("rows_var ",rows_var);
   Print("var value  ",matrix_var);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_var [10.5,9.1875,15.6875]
```

```
   rows_var [12.666667,20.666666,0.66666669,2.6666667]
   var value  11.916666984558105
   */
```

LinearRegression

Calcola un vettore/matrice con i valori di regressione lineare calcolati.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE      // asse lungo il quale viene calcolata la regr
   );
```

Parametri

axis

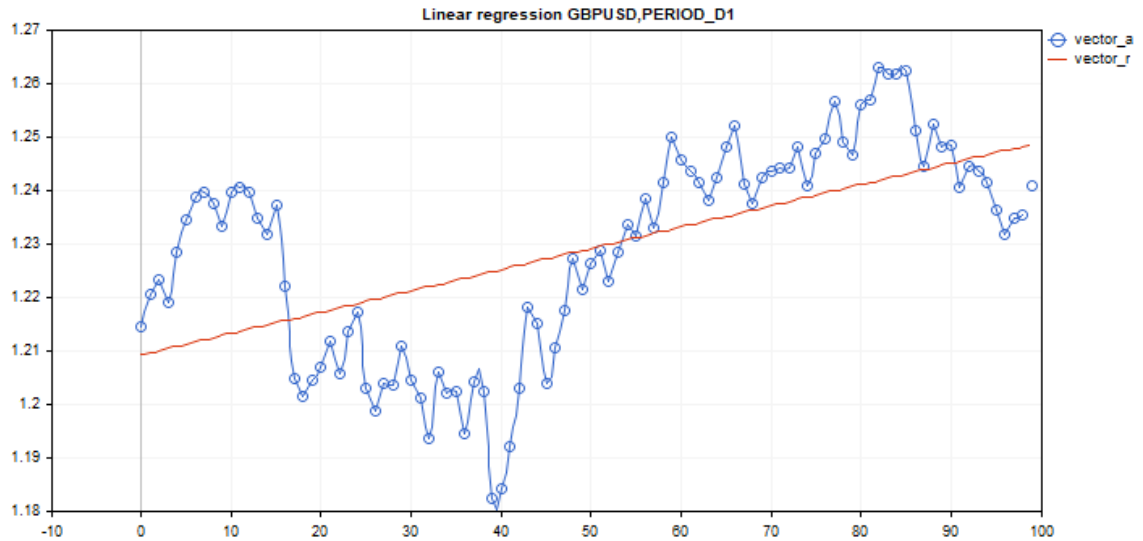
[in] Specificare l'asse lungo il quale viene calcolata la regressione. Valore dall'enumerazione ENUM_MATRIX_AXIS (AXIS_HORZ — asse orizzontale, AXIS_VERT — asse verticale).

Valore Restituito

Vettore o matrice con i valori di regressione lineare calcolati.

Nota

La regressione lineare è calcolata utilizzando l'equazione di regressione standard: y (x) = a * x + b, dove a è la pendenza della linea, mentre b è il suo spostamento sull'asse Y.



Esempio:

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

//+------------------------------------------------------------------+
```

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector vector_a;
   vector_a.CopyRates(_Symbol,_Period,COPY_RATES_CLOSE,1,100);
   vector vector_r=vector_a.LinearRegression();

//-- disattiva la visualizzazione del grafico
   ChartSetInteger(0,CHART_SHOW,false);

//-- array per disegnare un grafico
   double x[];
   double y1[];
   double y2[];
   ArrayResize(x,uint(vector_a.Size()));
   ArrayResize(y1,uint(vector_a.Size()));
   ArrayResize(y2,uint(vector_a.Size()));
   for(ulong i=0; i<vector_a.Size(); i++)
     {
      x[i]=(double)i;
      y1[i]=vector_a[i];
      y2[i]=vector_r[i];
     }

//--- titolo del grafico
   string title="Linear regression "+_Symbol+","+EnumToString(_Period);

   long   chart=0;
   string name="LinearRegression";

//-- crea un grafico
   CGraphic graphic;
   graphic.Create(chart,name,0,0,0,GRAPH_WIDTH,GRAPH_HEIGHT);
   graphic.BackgroundMain(title);
   graphic.BackgroundMainSize(12);

//-- grafico della funzione di attivazione
   CCurve *curvef=graphic.CurveAdd(x,y1,CURVE_POINTS_AND_LINES);
   curvef.Name("vector_a");
   curvef.LinesWidth(2);
   curvef.LinesSmooth(true);
   curvef.LinesSmoothTension(1);
   curvef.LinesSmoothStep(10);

//-- derivata della funzione di attivazione
   CCurve *curved=graphic.CurveAdd(x,y2,CURVE_LINES);
   curved.Name("vector_r");
   curved.LinesWidth(2);
```

```
   curved.LinesSmooth(true);
   curved.LinesSmoothTension(1);
   curved.LinesSmoothStep(10);
   graphic.CurvePlotAll();
   graphic.Update();

//--- loop infinito per riconoscere i pulsanti della tastiera premuti
   while(!IsStopped())
     {
      //--- premere il pulsante escape per uscire dal programma
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
         break;
      //--- premere PdDn per salvare l'immagine del grafico
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0)
        {
         string file_names[];
         if(FileSelectDialog("Save Picture",NULL,"All files (*.*)|*.*",FSD_WRITE_FILE,
            continue;
         ChartScreenShot(0,file_names[0],GRAPH_WIDTH,GRAPH_HEIGHT);
        }
      Sleep(10);
     }

//-- ripulire
   graphic.Destroy();
   ObjectDelete(chart,name);
   ChartSetInteger(0,CHART_SHOW,true);
  }
```

ENUM_MATRIX_AXIS

Enumerazione per specificare l'asse in tutte le funzioni statistiche per le matrici.

ID	Descrizione
AXIS_NONE	L'asse non è specificato. Il calcolo viene eseguito su tutti gli elementi della matrice, come se fosse un vettore (vedere il metodo Flat).
AXIS_HORZ	Asse orizzontale
AXIS_VERT	Asse verticale

Metodi sulle caratteristiche

Questi metodi consentono la ricezione delle caratteristiche della matrice, come:

- numero di righe
- numero di colonne
- norma
- numero di condizionamento
- determinante
- rango di una matrice
- segno
- spettro

Funzione	Azione
Rows	Restituisce il numero di righe in una matrice
Cols	Restituisce il numero di colonne in una matrice
Size	Restituisce la dimensione del vettore
Norm	Restituisce la norma di una matrice o vettore
Cond	Calcola il numero di condizionamento di una matrice
Det	Calcola il determinante di una matrice invertibile quadrata
SLogDet	Calcola il segno e il logaritmo del determinante di una matrice
Rank	Restituisce il rango di una matrice utilizzando il metodo Gaussiano
Trace	Restituisce la somma sulle diagonali della matrice
Spectrum	Calcola lo spettro di una matrice come l'insieme dei suoi autovalori dal prodotto AT*A

Rows

Restituisce il numero di righe in una matrice.

```
ulong matrix::Rows()
```

Valore Restituito

Intero.

Esempio

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Rows()=", m.Rows());
Print("m.Cols()=", m.Cols());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Rows()=3
 m.Cols()=4
*/
```

Cols

Restituisce il numero di colonne in una matrice.

```
ulong matrix::Cols()
```

Valore Restituito

Intero.

Esempio

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Cols()=", m.Cols());
Print("m.Rows()=", m.Rows());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Cols()=4
 m.Rows()=3
*/
```

Size

Restituisce la dimensione del vettore.

```
ulong vector::Size()
```

Valore Restituito

Intero.

Esempio

```
matrix m={{1,2,3,4,5,6,7,8,9,10,11,12}};
m.Reshape(3,4);
Print("matrix m\n",m);
vector v=m.Row(1);
Print("v.Size()=",v.Size());
Print("v=",v);


/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 v.Size()=4
 v=[5,6,7,8]
*/
```

Norm

Restituisce la norma di una matrice o vettore.

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,     // norma del vettore
  const int               norm_p=2  // numero della p-norma nel caso di VECTOR_NORM_P
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm      // norma matriciale
   );
```

Parametri

norma

[in] Ordine della norma

Valore Restituito

Norma della matrice o vettore

Note

- VECTOR_NORM_INF è il valore massimo assoluto tra gli elementi del vettore.
- VECTOR_NORM_MINUS_INF è il valore minimo assoluto di un vettore.
- VECTOR_NORM_P è la p-norma del vettore. Se norm_p=0, allora questo è il numero di elementi vettoriali diversi da zero. norm_p=1 è la somma dei valori assoluti degli elementi del vettore. norm_p=2 è la radice quadrata della somma dei quadrati dei valori degli elementi del vettore. Norma-p può essere negativa.
- MATRIX_NORM_FROBENIUS è la radice quadrata della somma dei quadrati dei valori degli elementi della matrice. La norma di Frobenius e la P2-norma vettore sono compatibili.
- MATRIX_NORM_SPECTRAL è il valore massimo dello spettro della matrice.
- MATRIX_NORM_NUCLEAR è la somma dei valori singolari della matrice.
- MATRIX_NORM_INF è la massima p1-norma vettore tra i vettori verticali della matrice. La matrice norm-inf e il vettore norm-inf sono compatibili.
- MATRIX_NORM_MINUS_INF è la minima p1_norma vettore tra i vettori verticali della matrice.
- MATRIX_NORM_P1 la massima p1-norma vettore tra i vettori orizzontali della matrice.
- MATRIX_NORM_MINUS_P1 è la minima p1_norma vettore tra i vettori orizzontali della matrice.
- MATRIX_NORM_P2 è il valore singolare più alto della matrice.
- MATRIX_NORM_MINUS_P2 è il valore singolare più basso di una matrice.

Un semplice algoritmo per calcolare la P-norma di un vettore in MQL5:

```
double VectorNormP(const vector& v,int norm_value)
  {
   ulong  i;
```

```
   double norm=0.0;
//---
   switch(norm_value)
     {
      case 0 :
         for(i=0; i<v.Size(); i++)
            if(v[i]!=0)
               norm+=1.0;
         break;
      case 1 :
         for(i=0; i<v.Size(); i++)
            norm+=MathAbs(v[i]);
         break;
      case 2 :
         for(i=0; i<v.Size(); i++)
            norm+=v[i]*v[i];
         norm=MathSqrt(norm);
         break;
      default :
         for(i=0; i<v.Size(); i++)
            norm+=MathPow(MathAbs(v[i]),norm_value);
         norm=MathPow(norm,1.0/norm_value);
     }
//---
   return(norm);
  }
```

Esempio in MQL5:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_FROBENIUS)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_INF)", b.Norm(MATRIX_NORM_INF));
  Print("b.Norm(MATRIX_NORM_MINUS_INF)", b.Norm(MATRIX_NORM_MINUS_INF));
  Print("b.Norm(MATRIX_NORM_P1)=)", b.Norm(MATRIX_NORM_P1));
  Print("b.Norm(MATRIX_NORM_MINUS_P1)=", b.Norm(MATRIX_NORM_MINUS_P1));
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_P2));
  Print("b.Norm(MATRIX_NORM_MINUS_P2)=", b.Norm(MATRIX_NORM_MINUS_P2));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
```

```
  [-1,0,1]
  [2,3,4]]
  b.Norm(MATRIX_NORM_P2)=7.745966692414834
  b.Norm(MATRIX_NORM_FROBENIUS)=7.745966692414834
  b.Norm(MATRIX_NORM_INF)9.0
  b.Norm(MATRIX_NORM_MINUS_INF)2.0
  b.Norm(MATRIX_NORM_P1)=)7.0
  b.Norm(MATRIX_NORM_MINUS_P1)=6.0
  b.Norm(MATRIX_NORM_P2)=7.348469228349533
  b.Norm(MATRIX_NORM_MINUS_P2)=1.857033188519056e-16
  */
```

Esempio in Python:

```
import numpy as np
from numpy import linalg as LA
a = np.arange(9) - 4
print("a \n",a)
b = a.reshape((3, 3))
print("b \n",b)
print("LA.norm(b)=",LA.norm(b))
print("LA.norm(b, 'fro')=",LA.norm(b, 'fro'))
print("LA.norm(b, np.inf)=",LA.norm(b, np.inf))
print("LA.norm(b, -np.inf)=",LA.norm(b, -np.inf))
print("LA.norm(b, 1)=",LA.norm(b, 1))
print("LA.norm(b, -1)=",LA.norm(b, -1))
print("LA.norm(b, 2)=",LA.norm(b, 2))
print("LA.norm(b, -2)=",LA.norm(b, -2))

a
 [-4 -3 -2 -1  0  1  2  3  4]
b
 [[-4 -3 -2]
 [-1  0  1]
 [ 2  3  4]]
LA.norm(b)= 7.745966692414834
LA.norm(b, 'fro')= 7.745966692414834
LA.norm(b, np.inf)= 9.0
LA.norm(b, -np.inf)= 2.0
LA.norm(b, 1)= 7.0
LA.norm(b, -1)= 6.0
LA.norm(b, 2)= 7.3484692283495345
LA.norm(b, -2)= 1.857033188519056e-16
```

Cond

Calcola il numero di condizionamento di una matrice.

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // norma matriciale
   );
```

Parametri

norma

[in] Ordine della norma da ENUM_MATRIX_NORM

Valore Restituito

Il numero di condizionamento di una matrice.Può essere infinito.

Note

Il numero di condizionamento di x è definito come la norma di x volte la norma dell'inversa di x [1]. La norma può essere la solita norma L2 (radice della somma dei quadrati) o una di una serie di altre norme matriciali.

Il numero di condizionamento è il valore K uguale al prodotto delle norme matriciali A e la sua inversa. Le matrici con un numero di condizionamento elevato sono chiamate mal condizionate. Quelle con un numero di condizionamento basso sono chiamate ben condizionate. La matrice inversa si ottiene mediante pseudo-inversione, in modo da non essere limitata dalla condizione di squadratura e di non-singolarità della matrice.

Un'eccezione è il numero di condizionamento spettrale.

Un semplice algoritmo per calcolare il numero di condizionamento spettrale in MQL5:

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
```

```
         }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

Esempio in MQL5:

```
  matrix a= {{1, 0, -1}, {0, 1, 0}, { 1, 0, 1}};
  Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
  Print("a.Cond(MATRIX_NORM_FROBENIUS)=", a.Cond(MATRIX_NORM_FROBENIUS));
  Print("a.Cond(MATRIX_NORM_INF)=", a.Cond(MATRIX_NORM_INF));
  Print("a.Cond(MATRIX_NORM_MINUS_INF)=", a.Cond(MATRIX_NORM_MINUS_INF));
  Print("a.Cond(MATRIX_NORM_P1)=)", a.Cond(MATRIX_NORM_P1));
  Print("a.Cond(MATRIX_NORM_MINUS_P1)=", a.Cond(MATRIX_NORM_MINUS_P1));
  Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
  Print("a.Cond(MATRIX_NORM_MINUS_P2)=", a.Cond(MATRIX_NORM_MINUS_P2));

  /*
  matrix a
  [[1,0,-1]
  [0,1,0]
  [1,0,1]]
  a.Cond(MATRIX_NORM_P2)=1.414213562373095
  a.Cond(MATRIX_NORM_FROBENIUS)=3.162277660168379
  a.Cond(MATRIX_NORM_INF)=2.0
  a.Cond(MATRIX_NORM_MINUS_INF)=0.9999999999999997
  a.Cond(MATRIX_NORM_P1)=)2.0
  a.Cond(MATRIX_NORM_MINUS_P1)=0.9999999999999998
  a.Cond(MATRIX_NORM_P2)=1.414213562373095
  a.Cond(MATRIX_NORM_MINUS_P2)=0.7071067811865472
  */
```

Esempio in Python:

```
import numpy as np
from numpy import linalg as LA
a = np.array([[1, 0, -1], [0, 1, 0], [1, 0, 1]])
print("a \n",a)
print("LA.cond(a)=",LA.cond(a))
print("LA.cond(a, 'fro')=",LA.cond(a, 'fro'))
print("LA.cond(a, np.inf)=",LA.cond(a, np.inf))
print("LA.cond(a, -np.inf)=",LA.cond(a, -np.inf))
```

```
print("LA.cond(a, 1)=",LA.cond(a, 1))
print("LA.cond(a, -1)=",LA.cond(a, -1))
print("LA.cond(a, 2)=",LA.cond(a, 2))
print("LA.cond(a, -2)=",LA.cond(a, -2))

a
 [[ 1  0 -1]
 [ 0  1  0]
 [ 1  0  1]]
LA.cond(a)= 1.4142135623730951
LA.cond(a, 'fro')= 3.1622776601683795
LA.cond(a, np.inf)= 2.0
LA.cond(a, -np.inf)= 1.0
LA.cond(a, 1)= 2.0
LA.cond(a, -1)= 1.0
LA.cond(a, 2)= 1.4142135623730951
LA.cond(a, -2)= 0.7071067811865475
```

Det

Calcola il determinante di una matrice invertibile quadrata.

```
double matrix::Det()
```

Valore Restituito

Determinante della matrice.

Note

I determinanti della matrice di ordine 2 e 3 sono calcolati secondo la regola di Sarrus. d2=a11*a22-a12*a21; d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33

Il determinante è calcolato con il metodo Gaussiano riducendo la matrice a una forma triangolare superiore. Il determinante di una matrice triangolare superiore è uguale al prodotto degli elementi delle diagonali principali.

Se almeno una riga o colonna della matrice è zero, il determinante è zero.

Se due o più righe o colonne della matrice sono dipendenti linearmente, il suo determinante è zero.

Il determinante di una matrice è uguale al prodotto dei suoi autovalori.

Esempio in MQL5:

```
matrix m={{1,2},{3,4}};
double det=m.Det();
Print("matrix m\n",m);
Print("det(m)=",det);
/*
matrix m
[[1,2]
 [3,4]]
det(m)=-2.0
*/
```

Esempio in Python:

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print('a \n',a)
print('nnp.linalg.det(a) \n',np.linalg.det(a))

a
 [[1 2]
 [3 4]]
```

```
np.linalg.det(a)
 -2.0000000000000004
```

SLogDet

Calcola il segno e il logaritmo di un determinante della matrice.

```
double matrix::SLogDet(
  int&  sign      // segno
   );
```

Parametri

segno

[out] Il segno del determinante. Se il segno è pari, il determinante è positivo.

Valore Restituito

Un numero che rappresenta il segno del determinante.

Note

Il determinante è calcolato con il metodo Gaussiano riducendo la matrice a una forma triangolare superiore. Il determinante di una matrice triangolare superiore è uguale al prodotto degli elementi delle diagonali principali. Il logaritmo di un prodotto è uguale alla somma dei logaritmi. Pertanto, in caso di overflow durante il calcolo del determinante, è possibile utilizzare il metodo SLogDet.

Se il segno è pari, il determinante è positivo.

Esempio

```
a = np.array([[1, 2], [3, 4]])  (sign, logdet) = np.linalg.slogdet(a)  (sign, logdet)
```

Rank

Restituisce il rango della matrice utilizzando il metodo Gaussiano.

```
int  Rank()
```

Valore Restituito

Rango della matrice.

Note

Il rango di un sistema di righe (o colonne) di una matrice A che ha m righe e n colonne è il numero massimo di righe (o colonne) indipendenti linearmente. Diverse righe (colonne) sono chiamate linearmente indipendenti se nessuna di esse può essere espressa linearmente in termini di altre. Il rango del sistema di righe è sempre uguale al rango del sistema di colonne. Questo valore è chiamato rango della matrice.

Esempio in MQL5:

```
matrix a=matrix::Eye(4, 4);;
Print("matrix a \n", a);
Print("a.Rank()=", a.Rank());

matrix I=matrix::Eye(4, 4);
I[3, 3] = 0.;    // rango di matrice insufficente
Print("I \n", I);
Print("I.Rank()=", I.Rank());

matrix b=matrix::Ones(1, 4);
Print("b \n", b);
Print("b.Rank()=", b.Rank());;// 1 dimensione - rango 1 salvo tutti 0

matrix  zeros=matrix::Zeros(4, 1);
Print("zeros \n", zeros);
Print("zeros.Rank()=", zeros.Rank());

/*
matrix a
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,1]]
a.Rank()=4

I
[[1,0,0,0]
[0,1,0,0]
```

```
  [0,0,1,0]
  [0,0,0,0]]
  I.Rank()=3

  b
  [[1,1,1,1]]
  b.Rank()=1

  zeros
  [[0]
  [0]
  [0]
  [0]]
  zeros.Rank()=0
  */
```

Esempio in Python:

```
import numpy as np
from numpy.linalg import matrix_rank
a=(np.eye(4)) # Full rank matrix
print("a \n", a)
print("matrix_rank(a)=",matrix_rank(a))
I=np.eye(4)
I[-1,-1] = 0. # rank deficient matrix
print("I \n",I)
print("matrix_rank(I)=",matrix_rank(I))

b=np.ones((4,))
print("b \n",b)
print("matrix_rank(b)=",matrix_rank(b)) # 1 dimension - rank 1 unless all 0

zeros=np.zeros((4,))
print("zeroes \n",zeros)
print("matrix_rank(zeros)=",matrix_rank(zeros))

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
matrix_rank(a)= 4

I
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
```

```
matrix_rank(I)= 3

b
 [1. 1. 1. 1.]
matrix_rank(b)= 1

zeroes
 [0. 0. 0. 0.]
matrix_rank(zeros)= 0
```

Trace

Restituisce la somma lungo le diagonali della matrice.

```
double matrix::Trace()
```

Valore Restituito

La somma lungo la diagonale.

Note

La traccia di una matrice è uguale alla somma dei suoi autovalori.

Esempio in MQL5:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a.Reshape(3, 3);
  Print("matrix a \n", a);
  Print("a.Trace() \n", a.Trace());

  /*
  matrix a
  [[0,1,2]
  [3,4,5]
  [6,7,8]]
  a.Trace()
  12.0
  */
```

Esempio in Python:

```
a = np.arange(9).reshape((3,3))
print('a \n',a)
print('np.trace(a) \n',np.trace(a))

a
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

np.trace(a)
 12
```

Spectrum

Calcola lo spettro di una matrice come l'insieme dei suoi autovalori dal prodotto AT*A.

```
vector matrix::Spectrum()
```

Valore Restituito

Spettro di una matrice come vettore di autovalori di matrice.

Esempio

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

Metodi matriciali per risolvere sistemi di equazioni lineari

Metodi per risolvere sistemi di equazioni lineari e calcolare la matrice inversa.

Funzione	Azione
Solve	Risolvere un'equazione lineare della matrice, o sistema di equazioni algebriche lineari
LstSq	Restituisce la soluzione dei minimi quadrati delle equazioni algebriche lineari (per matrici non quadrate o degenerate)
Inv	Calcola l'inversa (moltiplicativo) di una matrice quadrata non degenerata con il metodo di Jordan-Gauss
PInv	Calcola la pseudoinversa di una matrice con il metodo di Moore-Penrose

Solve

Risolvere un'equazione lineare della matrice, o sistema di equazioni algebriche lineari.

```
vector matrix::Solve(
  const vector  b      // valori 'variabile dipendente' o ordinata
   );
```

Parametri

b

[in] valori 'variabile dipendente' o ordinata (Vettore di termini liberi).

Valore Restituito

Vettore con soluzione al sistema a * x = b.

Note

Se almeno una riga o colonna della matrice è zero, il sistema non ha soluzione.

Se due o più righe o colonne della matrice sono dipendenti linearmente, il sistema non ha soluzione.

Esempio

```
//--- soluzione SLAE
   vector_x=matrix_a.Solve(vector_b);
//--- controlla se a * x = b
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

LstSq

Restituisce la soluzione dei minimi quadrati delle equazioni algebriche lineari (per matrici non quadrate o degenerate).

```
vector matrix::LstSq(
  const vector  b      // valori 'variabile dipendente' o ordinata
   );
```

Parametri

b

[in] valori 'variabile dipendente' o ordinata (Vettore di termini liberi)

Valore Restituito

Vettore con soluzione al sistema a * x = b. Questo è vero solo per i sistemi che hanno una soluzione esatta.

Esempio

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//-- controllo, deve essere [5, -4]
   Print("x=", x);
//-- controllo, deve essere [7, 40, 3]
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

Inv

Calcola l'inverso moltiplicativo di una matrice invertibile quadrata con il metodo di Jordan-Gauss.

```
matrix matrix::Inv()
```

Valore Restituito

Inverso moltiplicativo della matrice.

Note

Il prodotto della matrice originale e della matrice inversa è la matrice identità.

Se almeno una riga o colonna della matrice è zero, la matrice inversa non può essere ottenuta.

Se due o più righe o colonne della matrice sono dipendenti linearmente, la matrice inversa non può essere ottenuta.

Esempio

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//-- popolare la matrice quadrata
   MatrixTestFirst(matrix_a);
//-- saranno misurati i microsecondi
   ulong t1=GetMicrosecondCount();
//--- ottenere la matrice inversa
   matrix inverse=matrix_a.Inv();
//-- misurare
   ulong t2=GetMicrosecondCount();
//-- verificare la correttezza
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
         //--- devono essere tutti uno lungo la diagonale
         if(i==j)
            value=1.0;
         else
            value=0.0;
         if(MathClassify(identity[i][j])>FP_ZERO)
            errors++;
         else
```

```
            {
             if(identity[i][j]!=value)
               {
                double diff=MathAbs(identity[i][j]-value);
                //--- troppe moltiplicazioni e disposizioni, quindi ridurre la precisio
                if(diff>1e-9)
                   errors++;
               }
            }
         }
      }
//---
   double elapsed_time=double(t2-t1)/1000.0;
   printf("Inversion of matrix %d x %d  %s  errors=%d  time=%.3f ms",size_m,size_m,er
   return(errors);
  }
```

PInv

Calcola la pseudoinversa di una matrice con il metodo di Moore-Penrose.

```
matrix matrix::PInv()
```

Valore Restituito

La matrice pseudoinversa

Esempio

```
int TestPseudoInverse(const int size_m, const int size_k)
  {
   matrix matrix_a(size_m,size_k);
   matrix matrix_inverted(size_k,size_m);
   matrix matrix_temp;
   matrix matrix_a2;
//---riempire la matrice
   MatrixTestFirst(matrix_a);
//-- invertire
   matrix_inverted=matrix_a.PInv();
//-- verificare la correttezza
   int errors=0;
//--- A * A+ * A = A   (A+ è una pseudoinversa di A)
   matrix_temp=matrix_a.MatMul(matrix_inverted);
   matrix_a2=matrix_temp.MatMul(matrix_a);
   errors=(int)matrix_a.CompareByDigits(matrix_a2,10);

   printf("PseudoInversion %s matrix_size  %d x %d  errors=%d",errors==0?"passed":"fai
//---
   return(errors);
  }
```

Machine learning

Questi metodi sono utilizzati nell'apprendimento automatico.

La funzione di attivazione della rete neurale determina il valore di uscita di un neurone secondo la somma ponderata degli input. La selezione della funzione di attivazione ha un grande impatto sulle prestazioni della rete neurale. Diverse parti del modello (strati) possono utilizzare diverse funzioni di attivazione.

Oltre a tutte le funzioni di attivazione note, MQL5 offre anche i loro derivati. I derivati della funzione permettono un aggiornamento efficiente dei parametri del modello basati sull'errore ricevuto nell'apprendimento.

Una rete neurale mira a trovare un algoritmo che minimizza l'errore nell'apprendimento, per il quale viene utilizzata la funzione di perdita. Il valore della funzione di perdita indica quanto il valore previsto dal modello devia da quello reale. Diverse funzioni di perdita vengono utilizzate a seconda del problema. Per esempio, Mean Squared Error (MSE) è usata per problemi di regressione, e Binary Cross-Entropy (BCE) è usata per scopi di classificazione binaria.

Funzione	Azione
Activation	Calcola i valori della funzione di attivazione e li scrive nel vettore/matrice passati
Derivative	Calcola i valori derivati della funzione di attivazione e li scrive nel vettore/matrice passato
Loss	Calcola i valori della funzione di perdita e li scrive nel vettore/matrice passati
LossGradient	Calcola un vettore o una matrice di gradienti della funzione di perdita
RegressionMetric	Calcola la metrica di regressione come errore della deviazione dalla linea di regressione costruita sull'array di dati specificato
ConfusionMatrix	Calcola la matrice della confusione. Il metodo viene applicato al vettore dei valori previsti
ConfusionMatrixMultilabel	Calcola la matrice di confusione per ogni etichetta. Il metodo viene applicato al vettore dei valori previsti
ClassificationMetric	Calcola la metrica di classificazione per valutare la qualità dei dati previsti rispetto ai dati reali Il metodo viene applicato al vettore dei valori previsti
ClassificationScore	Calcola la metrica di classificazione per valutare la qualità dei dati previsti rispetto ai dati reali

Esempio

Questo esempio dimostra l'addestramento di un modello utilizzando operazioni matriciali. Il modello è addestrato per la funzione (a + b + c)^2 / (a^2 + b^2 + c^2). Inseriamo la matrice di dati iniziale, in cui a, b e c sono contenuti in colonne differenti. Il risultato della funzione viene ottenuto all'output del modello.

```
matrix weights1, weights2, weights3;                // matrici dei pesi
matrix output1, output2, result;                    // output dello strato neurale dell
input int layer1 = 200;                             // la dimensione del primo livello
input int layer2 = 200;                             // la dimensione del secondo livell
input int Epochs = 20000;                           //il numero dei periodi di addestra
input double lr = 3e-6;                             // tasso di apprendimento
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // funzione di attivazione
//+------------------------------------------------------------------+
//| Script start function                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;    // dimensione del campione di addestramento
   int test = 10;       // dimensione del campione di prova
   matrix m_data, m_target;
//-- generazione di un campione di addestramento
   if(!CreateData(m_data, m_target, train))
      return;
//-- addestrare il modello
   if(!Train(m_data, m_target, Epochs))
      return;
//-- genera un campione di prova
   if(!CreateData(m_data, m_target, test))
      return;
//-- testa il modello
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
// Metodo di generazione del campione                                |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- inizializza i dati iniziali e le matrici risultanti
   if(!data.Init(count, 3) || !target.Init(count, 1))
      return false;
//-- riempire la matrice dei dati iniziali con valori casuali
   data.Random(-10, 10);
//-- calcola i valori target per il campione di allenamento
   vector X1 = MathPow(data.Col(0) + data.Col(1) + data.Col(1), 2);
   vector X2 = MathPow(data.Col(0), 2) + MathPow(data.Col(1), 2) + MathPow(data.Col(2)
   if(!target.Col(X1 / X2, 0))
      return false;
//-- risultato restituito
   return true;
  }
//+------------------------------------------------------------------+
//| Metodo del modello di addestramento                              |
//+------------------------------------------------------------------+
```

```
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
//-- creare il modello
   if(!CreateNet())
      return false;
//-- addestrare il modello
   for(int ep = 0; ep < epochs; ep++)
     {
      //--- feedforward pass
      if(!FeedForward(data))
         return false;
      PrintFormat("Epoch %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
      //--- backpropagation e aggiornamento della matrice del peso
      if(!Backprop(data, target))
         return false;
     }
//-- risultato restituito
   return true;
  }
//+------------------------------------------------------------------+
//| Metodo del modello della creazione                               |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- inizializza le matrici del peso
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) || !weights3.Ini
      return false;
//-- riempire le matrici del peso con valori casuali
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//-- risultato restituito
   return true;
  }
//+------------------------------------------------------------------+
//| Metodo Feedforward                                               |
//+------------------------------------------------------------------+
bool FeedForward(matrix &data)
  {
//-- verifica la dimensione dei dati iniziali
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//-- calcola il primo strato neurale
   matrix temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   output1 = temp.MatMul(weights1);
//-- calcola la funzione di attivazione
```

```
   if(!output1.Activation(temp, ac_func))
      return false;
//-- calcola il secondo strato neurale
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//-- calcola la funzione di attivazione
   if(!output2.Activation(temp, ac_func))
      return false;
//-- calcola il terzo strato neurale
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//-- risultato restituito
   return true;
  }
//+------------------------------------------------------------------+
//| Metodo Backpropagation                                           |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//-- controllare la dimensione della matrice dei valori target
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//-- determinare la deviazione dei valori calcolati dall'obiettivo
   matrix loss = (target - result) * 2;
//-- propagare il gradiente al livello precedente
   matrix gradient = loss.MatMul(weights3.Transpose());
//-- aggiornare la matrice del peso dell'ultimo livello
   matrix temp;
   if(!output2.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   weights3 = weights3 + temp.Transpose().MatMul(loss) * lr;
//--- regola il gradiente di errore in base alla derivata della funzione di attivazior
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//-- propagare il gradiente ad uno strato inferiore
   gradient = loss.MatMul(weights2.Transpose());
//-- aggiornare la matrice di peso del secondo livello nascosto
   if(!output1.Activation(temp, ac_func))
```

```
      return false;
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   weights2 = weights2 + temp.Transpose().MatMul(loss) * lr;
//--- regola il gradiente di errore in base alla derivata della funzione di attivazion
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//-- aggiornare la matrice di peso del primo livello nascosto
   temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   weights1 = weights1 + temp.Transpose().MatMul(loss) * lr;
//-- risultato restituito
   return true;
  }
//+------------------------------------------------------------------+
//| Metodo di prova del modello                                      |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//-- feedforward sui dati di prova
   if(!FeedForward(data))
      return false;
//-- registra i risultati del calcolo del modello e i valori reali
   PrintFormat("Test loss %.5f", result.Loss(target, LOSS_MSE));
   ulong total = data.Rows();
   for(ulong i = 0; i < total; i++)
      PrintFormat("(%.2f + %.2f + %.2f)^2 / (%.2f^2 + %.2f^2 + %.2f^2) =  Net %.2f, Ta
                  data[i, 0], data[i, 1], data[i, 2], result[i, 0], target[i, 0]);
//-- risultato restituito
   return true;
  }
//+------------------------------------------------------------------+
```

Activation

Calcola i valori della funzione di attivazione e li scrive nel vettore/matrice passati.

```
bool vector::Activation(
  vector&                    vect_out,      // vettore per ottenere i valori
  ENUM_ACTIVATION_FUNCTION   activation,    // funzione di attivazione
   ...                                      // parametri addizionali
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // matrice per ottenere i valori
  ENUM_ACTIVATION_FUNCTION   activation      // funzione di attivazione
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // matrice per ottenere i valori
  ENUM_ACTIVATION_FUNCTION   activation,    // funzione di attivazione
  ENUM_MATRIX_AXIS           axis,          // asse
   ...                                      // parametri addizionali
   );
```

Parametri

vect_out/matrix_out

[out] Vettore o matrice per ottenere i valori calcolati della funzione di attivazione.

activation

[in] Funzione di attivazione dall'enumerazione ENUM_ACTIVATION_FUNCTION.

axis

[in] Valore dall'enumerazione ENUM_MATRIX_AXIS (AXIS_HORZ — asse orizzontale, AXIS_VERT — asse verticale).

...

[in] Parametri aggiuntivi richiesti per alcune funzioni di attivazione. Se non vengono specificati parametri, vengono utilizzati i valori predefiniti.

Valore Restituito

Restituisce true in caso di successo, altrimenti - false.

Parametri Addizionali

Alcune funzioni di attivazione accettano parametri addizionali. Se non vengono specificati parametri, vengono utilizzati i valori predefiniti

```
AF_ELU  (Exponential Linear Unit)
  double alpha=1.0

Activation function: if(x>=0) f(x) = x
                     else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

Activation function: f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit)
  double alpha=0.3

Activation function: if(x>=0) f(x)=x
                     else f(x) = alpha*x


AF_RELU  (REctified Linear Unit)
  double alpha=0.0
  double max_value=0.0
  double treshold=0.0

Activation function: if(alpha==0) f(x) = max(x,0)
                     else if(x>max_value) f(x) = x
                     else f(x) = alpha*(x - treshold)


AF_SWISH
  double beta=1.0

Activation function: f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit)
  double theta=1.0

Activation function: if(x>theta) f(x) = x
                     else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit)
  double alpha[] - learned array of coeefficients
```

```
Activation function: if(x[i]>=0) f(x)[i] = x[i]
                     else f(x)[i] = alpha[i] * x[i]
```

Nota

Nelle reti neurali artificiali, la funzione di attivazione di un neurone determina il segnale in uscita, che è definito da un segnale in ingresso o da un insieme di segnali in ingresso. La selezione della funzione di attivazione ha un grande impatto sulle prestazioni della rete neurale. Diverse parti del modello (strati) possono utilizzare funzioni di attivazione differenti.

Esempi di utilizzo di parametri addizionali:

```
vector x={0.1, 0.4, 0.9, 2.0, -5.0, 0.0, -0.1};
vector y;

x.Activation(y,AF_ELU);
Print(y);
x.Activation(y,AF_ELU,2.0);
Print(y);

Print("");
x.Activation(y,AF_LINEAR);
Print(y);
x.Activation(y,AF_LINEAR,2.0);
Print(y);
x.Activation(y,AF_LINEAR,2.0,5.0);
Print(y);

Print("");
x.Activation(y,AF_LRELU);
Print(y);
x.Activation(y,AF_LRELU,1.0);
Print(y);
x.Activation(y,AF_LRELU,0.1);
Print(y);

Print("");
x.Activation(y,AF_RELU);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5,1.0);
Print(y);

Print("");
```

```
   x.Activation(y,AF_SWISH);
   Print(y);
   x.Activation(y,AF_SWISH,2.0);
   Print(y);

   Print("");
   x.Activation(y,AF_TRELU);
   Print(y);
   x.Activation(y,AF_TRELU,0.3);
   Print(y);

   Print("");
   vector a=vector::Full(x.Size(),2.0);
   x.Activation(y,AF_PRELU,a);
   Print(y);

/*  Risultato
   [0.1,0.4,0.9,2,-0.993262053000915,0,-0.095162581964040]
   [0.1,0.4,0.9,2,-1.986524106001829,0,-0.190325163928081]

   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.2,0.8,1.8,4,-10,0,-0.2]
   [5.2,5.8,6.8,9,-5,5,4.8]

   [0.1,0.4,0.9,2,-1.5,0,-0.03]
   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.1,0.4,0.9,2,-0.5,0,-0.01]

   [0.1,0.4,0.9,2,0,0,0]
   [0.2,0.8,0.9,2,-10,0,-0.2]
   [-1.8,-1.2,0.9,2,-12,-2,-2.2]

   [0.052497918747894,0.239475064044981,0.6398545523625035,1.761594155955765,-0.033464
   [0.054983399731247,0.275989792451045,0.7723340415895611,1.964027580075817,-0.000226

   [0,0,0,2,0,0,0]
   [0,0.4,0.9,2,0,0,0]

   [0.1,0.4,0.9,2,-10,0,-0.2]
*/
```

Derivative

Calcola i valori della derivata della funzione di attivazione e li scrive nel vettore/matrice passato

```
bool vector::Derivative(
  vector&                    vect_out,      // vettore per ottenere i valori
  ENUM_ACTIVATION_FUNCTION   activation,    // funzione di attivazione
   ...                                      // parametri addizionali
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // matrice per ottenere i valori
  ENUM_ACTIVATION_FUNCTION   activation,    // funzione di attivazione
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // matrice per ottenere i valori
  ENUM_ACTIVATION_FUNCTION   activation,    // funzione di attivazione
  ENUM_MATRIX_AXIS           axis,          // asse
   ...                                      // parametri addizionali
   );
```

Parametri

vect_out/matrix_out

[out] Vettore o matrice per ottenere i valori calcolati della derivata della funzione di attivazione.

activation

[in] Funzione di attivazione dall'enumerazione ENUM_ACTIVATION_FUNCTION.

axis

[in] Valore dall'enumerazione ENUM_MATRIX_AXIS (AXIS_HORZ — asse orizzontale, AXIS_VERT — asse verticale).

...

[in] I parametri addizionali sono gli stessi delle funzioni di attivazione. Solo alcune funzioni di attivazione accettano parametri addizionali. Se non vengono specificati parametri, vengono utilizzati i valori predefiniti.

Valore Restituito

Restituisce true in caso di successo, altrimenti - false.

Nota

La funzione derivata consente un aggiornamento efficente dei parametri del modello basati sull'errore ricevuto nell'apprendimento durante la retropropagazione dell'errore.

Loss

Calcola il valore della funzione di perdita.

```
double vector::Loss(
  const vector&        vect_true,     // vettore di valori reali
  ENUM_LOSS_FUNCTION   loss,          // funzione di perdita
   ...                                // parametri addizionali
   );


double matrix::Loss(
  const matrix&        matrix_true,   // matrice di valori reali
  ENUM_LOSS_FUNCTION   loss,          // funzione di perdita
   );


double matrix::Loss(
  const matrix&        matrix_true,   // matrice di valori reali
  ENUM_LOSS_FUNCTION   loss,          // funzione di perdita
  ENUM_MATRIX_AXIS     axis,          // asse
   ...                                // parametri addizionali
   );
```

Parametri

vect_true/matrix_true

[in] Vettore o matrice di valori reali.

loss

[in] Funzione di perdita dall'enumerazione ENUM_LOSS_FUNCTION.

axis

[in] Valore dall'enumerazione ENUM_MATRIX_AXIS (AXIS_HORZ — asse orizzontale, AXIS_VERT — asse verticale).

...

[in] Parametro addizionale 'delta' può essere utilizzato solo dalla funzione di perdita Hubert (LOSS_HUBER)

Valore Restituito

Valore double.

Come viene utilizzato il parametro 'delta' nella funzione di perdita Hubert (LOSS_HUBER)

```
   double delta = 1.0;
   double error = fabs(y - x);
   if(error<delta)
      loss = 0.5 * error^2;
```

```
   else
      loss = 0.5 * delta^2 + delta * (error - delta);
```

Nota

Una rete neurale mira a trovare gli algoritmi che minimizzano l'errore sul campione di addestramento, per cui la funzione di perdita è utilizzata.

Il valore della funzione di perdita indica quanto il valore previsto dal modello devia da quello reale.

Differenti funzioni di perdita vengono utilizzate a seconda del problema. Per esempio, Mean Squared Error (MSE) è utilizzata per problemi di regressione, e Binary Cross-Entropy (BCE) è utilizzata per scopi di classificazione binaria.

Esempio di chiamata della funzione di perdita Hubert:

```
   vector y_true = {0.0, 1.0, 0.0, 0.0};
   vector y_pred = {0.6, 0.4, 0.4, 0.6};
   double loss=y_pred.Loss(y_true,LOSS_HUBER);
   Print(loss);
   double loss2=y_pred.Loss(y_true,LOSS_HUBER,0.5);
   Print(loss2);

/* Risultato
   0.155
   0.15125
*/
```

LossGradient

Calcola un vettore o una matrice di gradienti della funzione di perdita.

```
vector vector::LossGradient(
  const vector&       vect_true,     // vettore di valori reali
  ENUM_LOSS_FUNCTION  loss,          // tipo di funzione di perdita
   ...                               // parametri addizionali
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // matrice di valori reali
  ENUM_LOSS_FUNCTION  loss,          // funzione di perdita
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // matrice di valori reali
  ENUM_LOSS_FUNCTION  loss,          // funzione di perdita
  ENUM_MATRIX_AXIS    axis,          // asse
   ...                               // parametri addizionali
   );
```

Parametri

vect_true/matrix_true

[in] Vettore o matrice di valori reali.

loss

[in] Funzione di perdita dall'enumerazione ENUM_LOSS_FUNCTION.

axis

[in] Valore dall'enumerazione ENUM_MATRIX_AXIS (AXIS_HORZ — asse orizzontale, AXIS_VERT — asse verticale).

...

[in] Parametro addizionale 'delta' può essere utilizzato solo dalla funzione di perdita Hubert (LOSS_HUBER)

Valore Restituito

Vettore o matrice dei valori del gradiente della funzione di perdita. Il gradiente è la derivata parziale rispetto a dx (x è il valore previsto) della funzione di perdita in un dato punto.

Nota

I gradienti sono utilizzati nelle reti neurali per regolare i pesi della matrice di peso durante la retropropagazione, durante l'allenamento del modello.

Una rete neurale mira a trovare gli algoritmi che minimizzano l'errore sul campione di addestramento, per cui la funzione di perdita è utilizzata.

Differenti funzioni di perdita vengono utilizzate a seconda del problema. Per esempio, Mean Squared Error (MSE) è utilizzata per problemi di regressione, e Binary Cross-Entropy (BCE) è utilizzata per scopi di classificazione binaria.

Esempio di calcolo dei gradienti della funzione di perdita

```
   matrixf y_true={{ 1, 2, 3, 4 },
                   { 5, 6, 7, 8 },
                   { 9,10,11,12 }};
   matrixf y_pred={{ 1, 2, 3, 4 },
                   {11,10, 9, 8 },
                   { 5, 6, 7,12 }};
   matrixf loss_gradient =y_pred.LossGradient(y_true,LOSS_MAE);
   matrixf loss_gradienth=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_HORZ);
   matrixf loss_gradientv=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_VERT);
   Print("loss gradients\n",loss_gradient);
   Print("loss gradients on horizontal axis\n",loss_gradienth);
   Print("loss gradients on vertical axis\n",loss_gradientv);

/* Risultato
   loss gradients
   [[0,0,0,0]
    [0.083333336,0.083333336,0.083333336,0]
    [-0.083333336,-0.083333336,-0.083333336,0]]
   loss gradients on horizontal axis
   [[0,0,0,0]
    [0.33333334,0.33333334,0.33333334,0]
    [-0.33333334,-0.33333334,-0.33333334,0]]
   loss gradients on vertical axis
   [[0,0,0,0]
    [0.25,0.25,0.25,0]
    [-0.25,-0.25,-0.25,0]]
*/
```

RegressionMetric

Calcola la metrica di regressione per valutare la qualità dei dati previsti rispetto ai dati reali

```
double vector::RegressionMetric(
   const vector&           vector_true,   // vettore di valori reali
   ENUM_REGRESSION_METRIC  metric         // tipo di metrica
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // matrice di valori reali
   ENUM_REGRESSION_METRIC  metric         // tipo di metrica
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // matrice di valori reali
   ENUM_REGRESSION_METRIC  metric,        // tipo di metrica
   int                     axis           // asse
   );
```

Parametri

vector_true/matrix_true

[in] Vettore o matrice di valori reali.

metric

[in] Tipo di metrica dell'enumerazione ENUM_REGRESSION_METRIC.

axis

[in] Asse. 0 — asse orizzontale, 1 — asse verticale.

Valore Restituito

La metrica calcolata che valuta la qualità dei dati previsti rispetto ai dati reali.

Note

- REGRESSION_MAE — errore assoluto medio che rappresenta le differenze assolute tra i valori predetti e i valori reali corrispondenti
- REGRESSION_MSE — errore quadratico medio che rappresenta le differenze al quadrato tra i valori predetti e i corrispondenti valori reali
- REGRESSION_RMSE — radice quadrata di MSE
- REGRESSION_R2 - 1 — MSE(regressione) / MSE(media)
- REGRESSION_MAPE — MAE come percentuale
- REGRESSION_MSPE — MSE come percentuale
- REGRESSION_RMSLE — RMSE calcolato su scala logaritmica

Esempio:

```
   vector y_true = {3, -0.5, 2, 7};
   vector y_pred = {2.5, 0.0, 2, 8};
//---
   double mse=y_pred.RegressionMetric(y_true,REGRESSION_MSE);
   Print("mse=",mse);
//---
   double mae=y_pred.RegressionMetric(y_true,REGRESSION_MAE);
   Print("mae=",mae);
//---
   double r2=y_pred.RegressionMetric(y_true,REGRESSION_R2);
   Print("r2=",r2);

  /* Risultato
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

ConfusionMatrix

Calcola la matrice della confusione. Il metodo viene applicato al vettore dei valori previsti.

```
matrix vector::ConfusionMatrix(
   const vector&        vect_true      // vettore di valori reali
   );


matrix vector::ConfusionMatrix(
   const vector&        vect_true,     // vettore di valori reali
   uint                 label          // valore dell'etichetta
   );
```

Parametri

vect_true

[in] Vettore di valori reali.

label

[in] Valore dell'etichetta per il calcolo della matrice di confusione.

Valore Restituito

Matrice di confusione. Se il valore dell'etichetta non è specificato, viene restituita una matrice di confusione multi-classe, dove ogni etichetta è abbinata l'una all'altra singolarmente. Se viene specificato un valore di etichetta, viene restituita una matrice 2 x 2, in cui l'etichetta specificata viene considerata positiva, mentre tutte le altre etichette sono negative (ovr, one vs rest).

Nota

La matrice di confusione C è tale che Cij è uguale al numero di osservazioni note nel gruppo i e che si prevede siano nel gruppo j. Così, nella classificazione binaria, il conteggio dei veri negativi (TN) è C00, falsi negativi (FN) è C10, veri positivi (TP) è C11 e falsi positivi (FP) è C01.

In altre parole, la matrice può essere rappresentata graficamente così:

TN	FP
FN	TP

Le dimensioni del vettore dei valori reali e del vettore dei valori previsti dovrebbero essere le stesse.

Esempio:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
```

```
   matrix confusion=y_pred.ConfusionMatrix(y_true);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,0);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,1);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,2);
   Print(confusion);


/*
  [[3,0,0,0,0,0,0,0,0,0]
   [0,3,0,0,0,0,0,0,0,0]
   [0,0,1,0,1,0,0,1,0,0]
   [0,0,0,1,0,0,0,1,0,0]
   [0,0,1,0,3,0,0,0,0,1]
   [0,0,0,0,0,2,0,0,0,0]
   [1,0,0,0,0,1,1,0,0,0]
   [0,0,0,0,0,0,0,2,0,1]
   [0,0,1,0,0,0,0,0,0,1]
   [0,0,0,0,0,0,0,0,0,4]]
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
*/
```

ConfusionMatrixMultilabel

Calcola la matrice di confusione per ogni etichetta. Il metodo viene applicato al vettore dei valori previsti.

```
uint vector::ConfusionMatrixMultiLabel(
   const vector&       vect_true,     // vettore di valori reali
   matrix&             confusions[]   // array delle matrici di confusione calcolate
   );
```

Parametri

vect_true

[in] Vettore di valori reali.

confusions

[out] Un array di matrici 2 x 2 con matrici di confusione calcolate per ogni etichetta.

Valore Restituito

Dimensione dell'array delle matrici di confusione calcolate. In caso di fallimento, restituisce 0

Nota

L'array risultante può essere dinamico o statico. Se l'array è statico, allora non deve avere dimensioni inferiori al numero di classi.

Le dimensioni del vettore dei valori reali e del vettore dei valori previsti dovrebbero essere le stesse.

Esempio:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix label_confusions[12];

   uint   res=y_pred.ConfusionMatrixMultiLabel(y_true,label_confusions);
   Print("res=",res,"  size=",label_confusions.Size());
   for(uint i=0; i<res; i++)
      Print(label_confusions[i]);


/*
  res=10  size=12
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
```

```
   [[25,2]
    [2,1]]
   [[28,0]
    [1,1]]
   [[24,1]
    [2,3]]
   [[27,1]
    [0,2]]
   [[27,0]
    [2,1]]
   [[25,2]
    [1,2]]
   [[28,0]
    [2,0]]
   [[23,3]
    [0,4]]
*/
```

ClassificationMetric

Calcola la metrica di classificazione per valutare la qualità dei dati previsti rispetto ai dati reali Il metodo viene applicato al vettore dei valori previsti.

```
vector vector::ClassificationMetric(
   const vector&               vect_true,     // vettore di valori reali
   ENUM_CLASSIFICATION_METRIC metric         // tipo di metrica
   );


vector vector::ClassificationMetric(
   const vector&               vect_true,     // vettore di valori reali
   ENUM_CLASSIFICATION_METRIC metric         // tipo di metrica
   ENUM_AVERAGE_MODE           mode           // modalità di calcolo della media
   );
```

Parametri

vect_true

[in] Vettore di valori reali.

metric

[in] Tipo di metrica dall'enumerazione ENUM_CLASSIFICATION_METRIC. Vengono applicati valori diversi da CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION e CLASSIFICATION_ROC_AUC (utilizzati nel metodo ClassificationScore).

mode

[in] Modalità di media dall'enumerazione ENUM_AVERAGE_MODE . Utilizzato per le metriche CLASSIFICATION_F1, CLASSIFICATION_JACCARD, CLASSIFICATION_PRECISION e CLASSIFICATION_RECALL .

Valore Restituito

Vettore contenente la metrica calcolata. Nel caso della modalità di media AVERAGE_NONE, il vettore contiene valori metrici per ciascuna classe senza calcolare la media. (Per esempio, nel caso della classificazione binaria, si tratterebbe di due metriche rispettivamente per 'false' e 'true').

Nota sulle modalità di media

AVERAGE_BINARY è significativo solo per la classificazione binaria.

AVERAGE_MICRO — calcola le metriche globalmente contando i veri positivi, i falsi negativi e i falsi positivi totali.

AVERAGE_MACRO — calcola le metriche per ogni etichetta e trova la loro media non ponderata. Questo non tiene conto dello sbilanciamento delle etichette.

AVERAGE_WEIGHTED — calcola le metriche per ogni etichetta e trova la loro media ponderata in base al supporto (il numero di istanze vere per ogni etichetta). Ciò altera la “macro” per tenere conto dello sbilanciamento delle etichette; può risultare in un punteggio F-score che non è compreso tra precisione e richiamo.

Esempio:

```
vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

vector accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
Print("accuracy=",accuracy);
vector balanced=y_pred.ClassificationMetric(y_true,CLASSIFICATION_BALANCED_ACCURACY
Print("balanced=",balanced);
Print("");

vector f1_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MICRO)
Print("f1_micro=",f1_micro);
vector f1_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MACRO)
Print("f1_macro=",f1_macro);
vector f1_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_WE
Print("f1_weighted=",f1_weighted);
vector f1_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_NONE);
Print("f1_none=",f1_none);
Print("");

vector jaccard_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVE
Print("jaccard_micro=",jaccard_micro);
vector jaccard_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVE
Print("jaccard_macro=",jaccard_macro);
vector jaccard_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,A
Print("jaccard_weighted=",jaccard_weighted);
vector jaccard_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERA
Print("jaccard_none=",jaccard_none);
Print("");

vector precision_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_micro=",precision_micro);
vector precision_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_macro=",precision_macro);
vector precision_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISI
Print("precision_weighted=",precision_weighted);
vector precision_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,A
Print("precision_none=",precision_none);
Print("");

vector recall_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_micro=",recall_micro);
vector recall_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_macro=",recall_macro);
vector recall_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVE
Print("recall_weighted=",recall_weighted);
```

```
   vector recall_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE
   Print("recall_none=",recall_none);
   Print("");

//--- classificazione binaria
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};

   vector f1_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_F1,AVERAGE_
   Print("f1_bin=",f1_bin);
   vector jaccard_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_JACCAR
   Print("jaccard_bin=",jaccard_bin);
   vector precision_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_PREC
   Print("precision_bin=",precision_bin);
   vector recall_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_RECALL,
   Print("recall_bin=",recall_bin);


/*
  accuracy=[0.6666666666666666]
  balanced=[0.6433333333333333]

  f1_micro=[0.6666666666666666]
  f1_macro=[0.6122510822510823]
  f1_weighted=[0.632049062049062]
  f1_none=[0.8571428571428571,1,0.3333333333333333,0.6666666666666666,0.6666666666666

  jaccard_micro=[0.5]
  jaccard_macro=[0.4921428571428572]
  jaccard_weighted=[0.5056349206349205]
  jaccard_none=[0.75,1,0.2,0.5,0.5,0.6666666666666666,0.3333333333333333,0.4,0,0.5714

  precision_micro=[0.6666666666666666]
  precision_macro=[0.6571428571428571]
  precision_weighted=[0.6706349206349207]
  precision_none=[0.75,1,0.3333333333333333,1,0.75,0.6666666666666666,1,0.5,0,0.571428

  recall_micro=[0.6666666666666666]
  recall_macro=[0.6433333333333333]
  recall_weighted=[0.6666666666666666]
  recall_none=[1,1,0.3333333333333333,0.5,0.6,1,0.3333333333333333,0.6666666666666666,

  f1_bin=[0.4444444444444445]
  jaccard_bin=[0.2857142857142857]
  precision_bin=[0.5]
  recall_bin=[0.4]
*/
```

ClassificationScore

Calcola la metrica di classificazione per valutare la qualità dei dati previsti rispetto ai dati reali.

A differenza di altri metodi nella sezione Machine Learning, questo si applica al vettore dei valori reali piuttosto che al vettore dei valori previsti.

```
vector vector::ClassificationScore(
   const matrix&               pred_scores,   // matrice contenente la distribuzione di
   ENUM_CLASSIFICATION_METRIC metric         // tipo di metrica
   ENUM_AVERAGE_MODE          mode           // modalità di calcolo della media
   );


vector vector::ClassificationScore(
   const matrix&               pred_scores,   // matrice contenente la distribuzione di
   ENUM_CLASSIFICATION_METRIC metric         // tipo di metrica
   int                        param          // parametri aggiuntivi
   );
```

Parametri

pred_scores

[in] Matrice contenente un insieme di vettori orizzontali con le probabilità per ciascuna classe. Il numero di righe della matrice dovrebbe corrispondere alla dimensione del vettore dei valori reali.

metric

[in] Tipo di metrica dall'enumerazione ENUM_CLASSIFICATION_METRIC. Vengono utilizzati i valori CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION e CLASSIFICATION_ROC_AUC.

mode

[in] Modalità di media dall'enumerazione ENUM_AVERAGE_MODE . Utilizzato per le metriche CLASSIFICATION_AVERAGE_PRECISION e CLASSIFICATION_ROC_AUC.

param

[in] Nel caso della metrica CLASSIFICATION_TOP_K_ACCURACY dovrebbe essere specificato il valore intero K invece della modalità di calcolo della media.

Valore Restituito

Vettore contenente la metrica calcolata. Nel caso della modalità di media AVERAGE_NONE, il vettore contiene valori metrici per ciascuna classe senza calcolare la media. (Per esempio, nel caso della classificazione binaria, si tratterebbe di due metriche rispettivamente per 'false' e 'true').

Nota sulle modalità di media

AVERAGE_BINARY è significativo solo per la classificazione binaria.

AVERAGE_MICRO— calcola le metriche globalmente considerando ogni elemento della matrice degli indicatori di classificazione come un'unica classe. La matrice degli indicatori di classificazione fa riferimento ad una matrice con un insieme di probabilità per ciascuna classe.

AVERAGE_MACRO — calcola le metriche per ogni etichetta e trova la loro media non ponderata. Questo non tiene conto dello sbilanciamento delle etichette.

AVERAGE_WEIGHTED — calcola le metriche per ogni etichetta e trova la loro media ponderata in base al supporto (il numero di istanze vere per ogni etichetta).

Nota

In caso di classificazione binaria, possiamo inserire non solo una matrice n x 2, dove la prima colonna contiene probabilità per un'etichetta negativa, e la seconda colonna contiene probabilità per un'etichetta positiva, ma anche una matrice costituita da una colonna con probabilità positive. Questo perché i modelli di classificazione binaria possono restituire due probabilità o una probabilità per un'etichetta positiva.

Esempio:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   //vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

//--- label scores          0         1         2         3         4         5
   matrix y_scores={{0.000109, 0.000186, 0.000449, 0.000052, 0.000002, 0.000022, 0.000
                    {0.000091, 0.081956, 0.916816, 0.001106, 0.000006, 0.000002, 0.000
                    {0.000108, 0.972863, 0.003600, 0.000021, 0.010479, 0.000015, 0.000
                    {0.925425, 0.000080, 0.002913, 0.000057, 0.000274, 0.000638, 0.061
                    {0.000060, 0.000126, 0.000006, 0.000000, 0.993513, 0.000000, 0.000
                    {0.000016, 0.982124, 0.000045, 0.000002, 0.008445, 0.000001, 0.000
                    {0.000000, 0.000040, 0.000001, 0.000000, 0.989395, 0.000167, 0.000
                    {0.000795, 0.002938, 0.023447, 0.007418, 0.021838, 0.002476, 0.000
                    {0.000091, 0.000226, 0.000038, 0.000007, 0.000048, 0.854910, 0.068
                    {0.000000, 0.000000, 0.000000, 0.000000, 0.003004, 0.000000, 0.000
                    {0.998856, 0.000009, 0.000976, 0.000002, 0.000000, 0.000013, 0.000
                    {0.000178, 0.000446, 0.000326, 0.000033, 0.000193, 0.000071, 0.998
                    {0.000005, 0.000016, 0.000153, 0.000045, 0.004110, 0.000012, 0.000
                    {0.994188, 0.000003, 0.002584, 0.000005, 0.000005, 0.000100, 0.000
                    {0.000173, 0.990569, 0.000792, 0.000040, 0.001798, 0.000035, 0.000
                    {0.000000, 0.000537, 0.000008, 0.005080, 0.000046, 0.992910, 0.000
                    {0.000127, 0.000003, 0.000003, 0.000000, 0.001583, 0.000000, 0.000
                    {0.000001, 0.000012, 0.000072, 0.000020, 0.000000, 0.000000, 0.000
                    {0.000020, 0.000105, 0.001139, 0.901343, 0.002132, 0.083873, 0.000
                    {0.000002, 0.000048, 0.000019, 0.000000, 0.999347, 0.000002, 0.000
                    {0.000059, 0.001344, 0.612502, 0.002749, 0.000229, 0.000678, 0.000
                    {0.000586, 0.000740, 0.001625, 0.000007, 0.269341, 0.000076, 0.016
                    {0.009547, 0.018055, 0.283795, 0.071079, 0.426074, 0.082335, 0.036
                    {0.002506, 0.002545, 0.001148, 0.005659, 0.020416, 0.000112, 0.006
                    {0.001263, 0.001769, 0.000293, 0.000011, 0.000302, 0.881768, 0.112
                    {0.002904, 0.002909, 0.013421, 0.001461, 0.007519, 0.001251, 0.000
```

```
                     {0.000055, 0.001080, 0.893158, 0.000000, 0.104492, 0.000159, 0.001
                     {0.000344, 0.002693, 0.071184, 0.000262, 0.000001, 0.000003, 0.000
                     {0.001404, 0.009375, 0.002638, 0.229189, 0.000064, 0.000896, 0.007
                     {0.491140, 0.000125, 0.000024, 0.000302, 0.000038, 0.034947, 0.473

   vector top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,1);
   Print("top 1 accuracy score = ",top_k);
   top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top 2 accuracy score = ",top_k);
   vector y_true2={0, 1, 2, 2};
   matrix y_score2={{0.5, 0.2, 0.2},  // 0 è in cima 2
                    {0.3, 0.4, 0.2},  // 1 è in cima 2
                    {0.2, 0.4, 0.3},  // 2 è in cima 2
                    {0.7, 0.2, 0.1}}; // 2 non è in cima 2
   top_k=y_true2.ClassificationScore(y_score2,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top k = ",top_k);
   Print("");

   vector ap_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score micro = ",ap_micro);
   vector ap_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score macro = ",ap_macro);
   vector ap_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PREC
   Print("average precision score weighted = ",ap_weighted);
   vector ap_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION
   Print("average precision score none = ",ap_none);
   Print("");

   vector area_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score micro = ",area_micro);
   vector area_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score macro = ",area_macro);
   vector area_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score weighted = ",area_weighted);
   vector area_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE
   Print("roc auc score none = ",area_none);
   Print("");

//--- classificazione binaria
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};
   vector y_score_true={0.3,0.7,0.1,0.6,0.9,0.0,0.4,0.2,0.8};
   matrix y_score1_bin(y_score_true.Size(),1);
   y_score1_bin.Col(y_score_true,0);
   matrix y_scores_bin={{0.7, 0.3},
                        {0.3, 0.7},
                        {0.9, 0.1},
                        {0.4, 0.6},
                        {0.1, 0.9},
```

```
                          {1.0, 0.0},
                          {0.6, 0.4},
                          {0.8, 0.2},
                          {0.2, 0.8}};

   vector ap=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECIS
   Print("average precision score binary = ",ap);
   vector ap2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_AVERAGE_PREC
   Print("average precision score binary = ",ap2);
   vector ap3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PREC
   Print("average precision score none = ",ap3);
   Print("");

   vector area=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score binary = ",area);
   vector area2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_ROC_AUC,AV
   Print("roc auc score binary = ",area2);
   vector area3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AV
   Print("roc auc score none = ",area3);


/*
  top 1 accuracy score = [0.6666666666666666]
  top 2 accuracy score = [1]
  top k = [0.75]

  average precision score micro = [0.8513333333333333]
  average precision score macro = [0.9326666666666666]
  average precision score weighted = [0.9333333333333333]
  average precision score none = [1,1,0.7,1,0.9266666666666666,0.8333333333333333,1,0.

  roc auc score micro = [0.9839506172839506]
  roc auc score macro = [0.9892068783068803]
  roc auc score weighted = [0.9887354497354497]
  roc auc score none = [1,1,0.9506172839506173,1,0.984,0.9821428571428571,1,0.9753086

  average precision score binary = [0.7961904761904761]
  average precision score binary = [0.7961904761904761]
  average precision score none = [0.7678571428571428,0.7961904761904761]

  roc auc score binary = [0.7]
  roc auc score binary = [0.7]
  roc auc score none = [0.7,0.7]
*/
```

PrecisionRecall

Compute values to construct a precision-recall curve. Similarly to ClassificationScore, this method is applied to the vector of true values.

```
bool vector::PrecisionRecall(
   const matrix&                 pred_scores,   // matrix containing the probability
   const ENUM_ENUM_AVERAGE_MODE  mode           // averaging mode
   matrix&                       precision,     // calculated precision values for ea
   matrix&                       recall,        // calculated recall values for each
   matrix&                       thresholds,    // threshold values sorted in descend
   );
```

Parameters

pred_scores

[in] A matrix containing a set of horizontal vectors with probabilities for each class. The number of matrix rows must correspond to the size of the vector of true values.

mode

[in] Averaging mode from the ENUM_AVERAGE_MODE enumeration. Only AVERAGE_NONE, AVERAGE_BINARY and AVERAGE_MICRO are used.

precision

[out] A matrix with calculated precision curve values. If no averaging is applied (AVERAGE_NONE), the number of rows in the matrix corresponds to the number of model classes. The number of columns corresponds to the size of the vector of true values (or the number of rows in the probability distribution matrix pred_score). In the case of microaveraging, the number of rows in the matrix corresponds to the total number of threshold values, excluding duplicates.

recall

[out] A matrix with calculated recall curve values.

threshold

[out] Threshold matrix obtained by sorting the probability matrix

Note

See notes for the ClassificationScore method.

Example

An example of collecting statistics from the mnist.onnx model (99% accuracy).

```
//--- data for classification metrics
   vectorf y_true(images);
   vectorf y_pred(images);
   matrixf y_scores(images,10);
//--- input-output
   matrixf image(28,28);
```

```
   vectorf result(10);

//--- testing
   for(int test=0; test<images; test++)
     {
      image=test_data[test].image;
      if(!OnnxRun(model,ONNX_DEFAULT,image,result))
        {
         Print("OnnxRun error ",GetLastError());
         break;
        }
      result.Activation(result,AF_SOFTMAX);
      //--- collect data
      y_true[test]=(float)test_data[test].label;
      y_pred[test]=(float)result.ArgMax();
      y_scores.Row(result,test);
     }    }
```

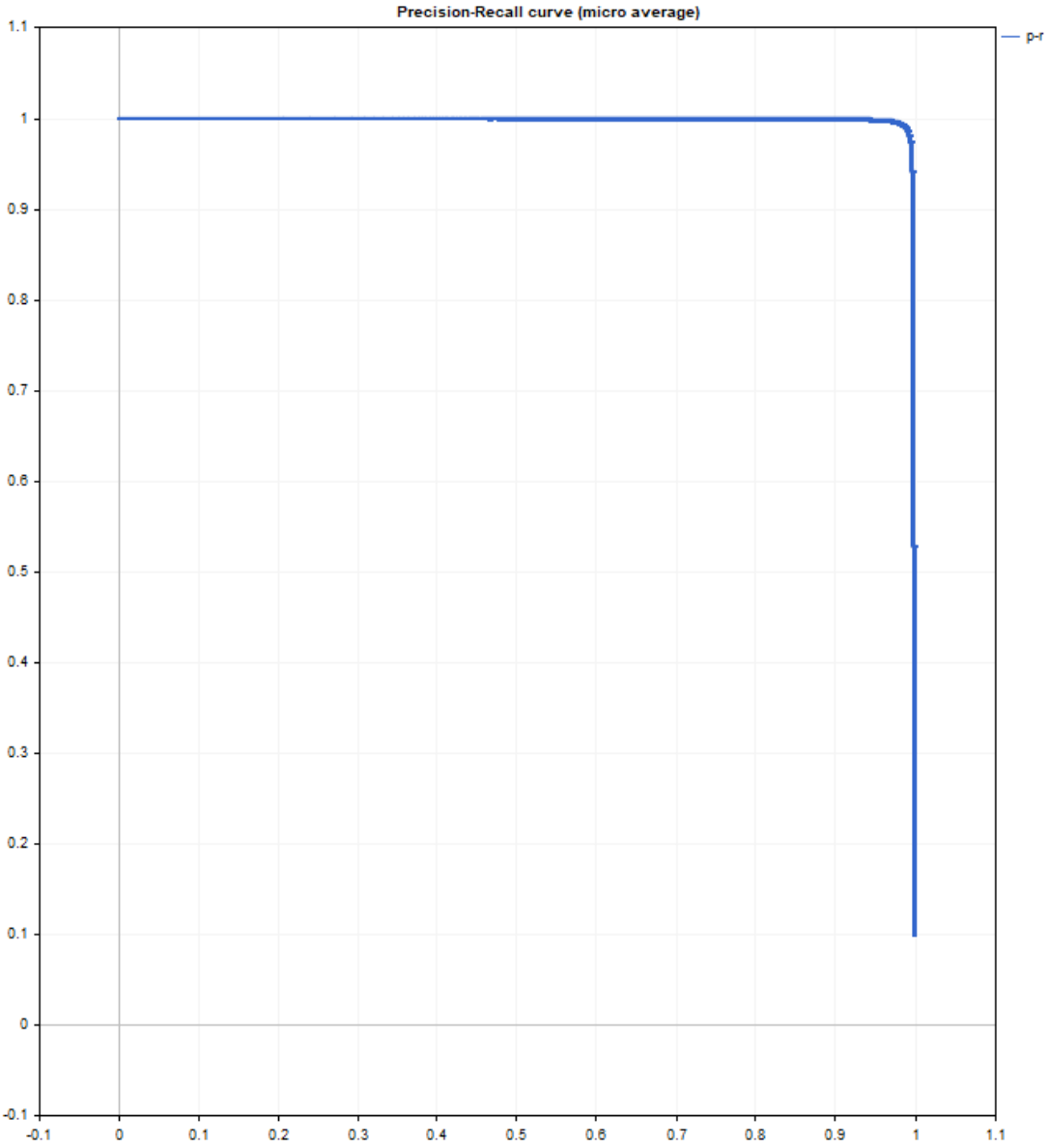
Accuracy calculation

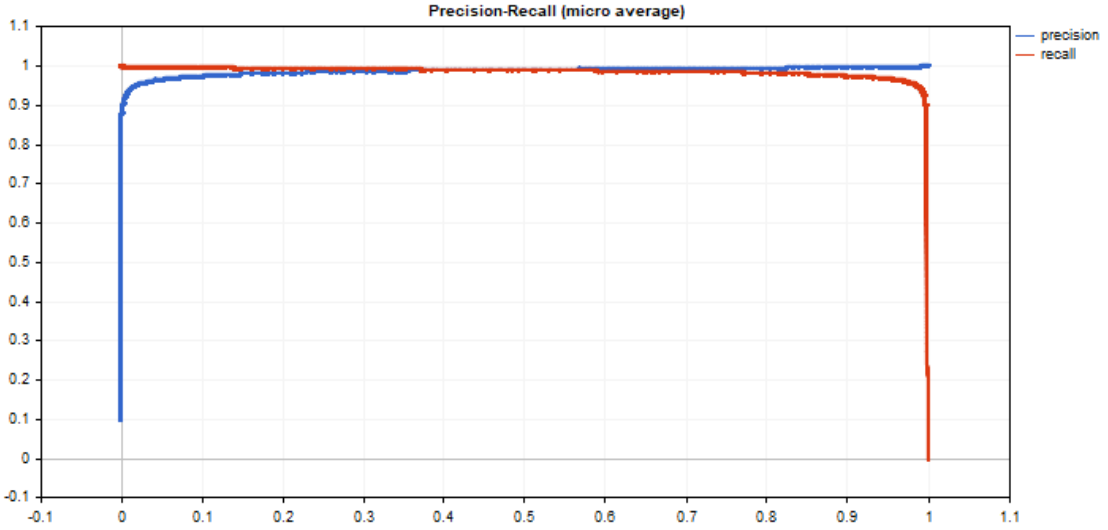
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

accuracy=0.989000
```

An example of plotting precision-recall graphs, where precision values are plotted on the y-axis and recall values are plotted on the x-axis. Also precision and recall graphs are plotted separately, with threshold values plotted on the x-axis

```
   if(y_true.PrecisionRecall(y_scores,AVERAGE_MICRO,mat_precision,mat_recall,mat_thres
     {
      double precision[],recall[],thres[];
      ArrayResize(precision,mat_thres.Cols());
      ArrayResize(recall,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<thres.Size(); i++)
        {
         precision[i]=mat_precision[0][i];
         recall[i]=mat_recall[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("Precision-Recall curve (micro average)","p-r","",recall,precision);
      Plot2Curves("Precision-Recall (micro average)","precision","recall",thres,precis
     }
```

Resulting curves:





ReceiverOperatingCharacteristic

Compute values to construct the Receiver Operating Characteristic (ROC) curve. Similarly to ClassificationScore, this method is applied to the vector of true values.

```
bool vector::ReceiverOperatingCharacteristic(
   const matrix&                 pred_scores,   // matrix containing the probability c
   const ENUM_ENUM_AVERAGE_MODE  mode           // averaging mode
   matrix&                       fpr,           // calculated false positive rate valu
   matrix&                       tpr,           // calculated true positive rate value
   matrix&                       thresholds,    // threshold values sorted in descendi
   );
```

Parameters

pred_scores

[in] A matrix containing a set of horizontal vectors with probabilities for each class. The number of matrix rows must correspond to the size of the vector of true values.

mode

[in] Averaging mode from the ENUM_AVERAGE_MODE enumeration. Only AVERAGE_NONE, AVERAGE_BINARY and AVERAGE_MICRO are used.

fpr

[out] A matrix with calculated values of the false positive rate curve. If no averaging is applied (AVERAGE_NONE), the number of rows in the matrix corresponds to the number of model classes. The number of columns corresponds to the size of the vector of true values (or the number of rows in the probability distribution matrix pred_score). In the case of microaveraging, the number of rows in the matrix corresponds to the total number of threshold values, excluding duplicates.

tpr

[out] A matrix with calculated values of the true positive rate curve.

threshold

[out] Threshold matrix obtained by sorting the probability matrix

Note

See notes for the ClassificationScore method.

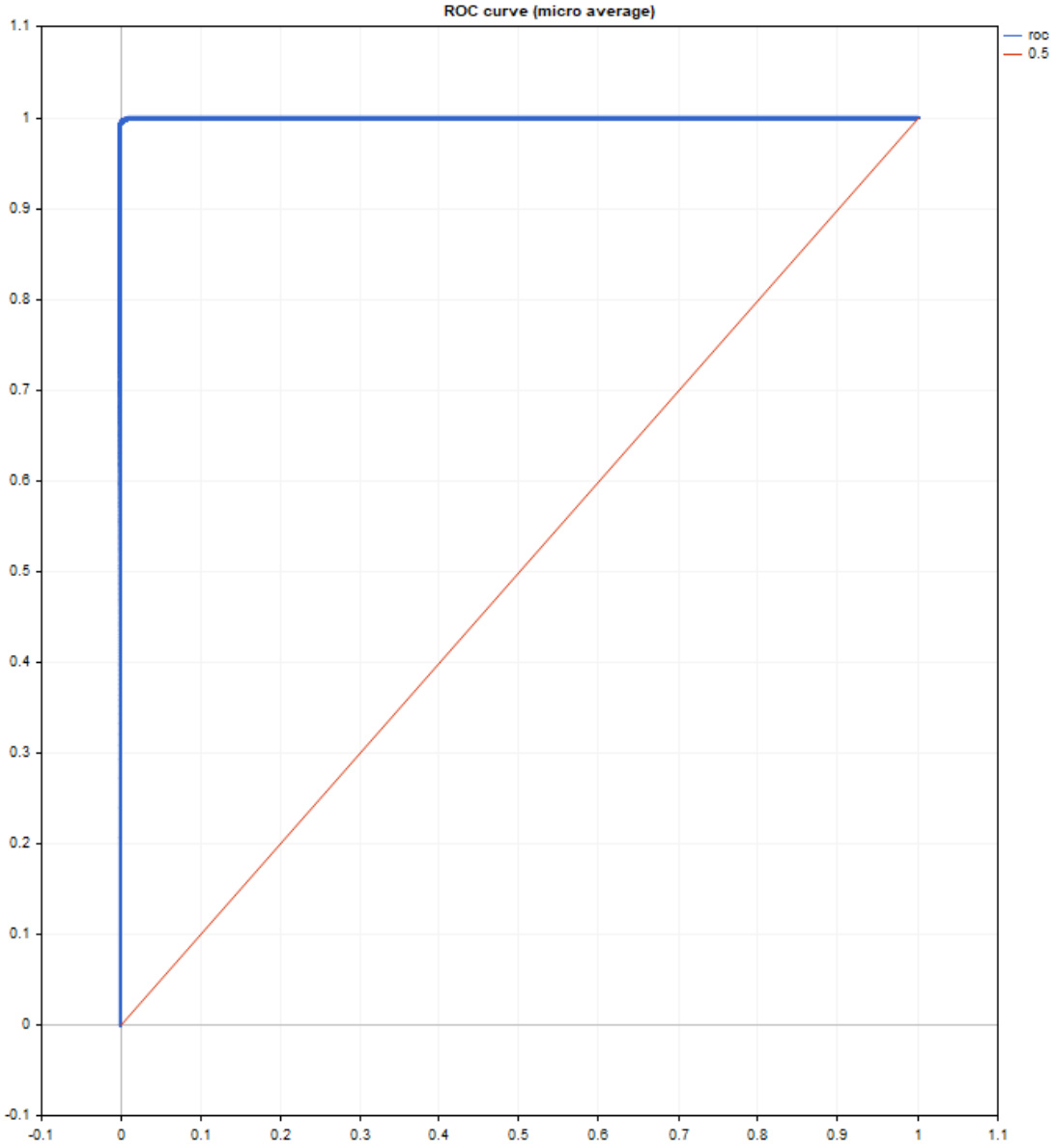
Example

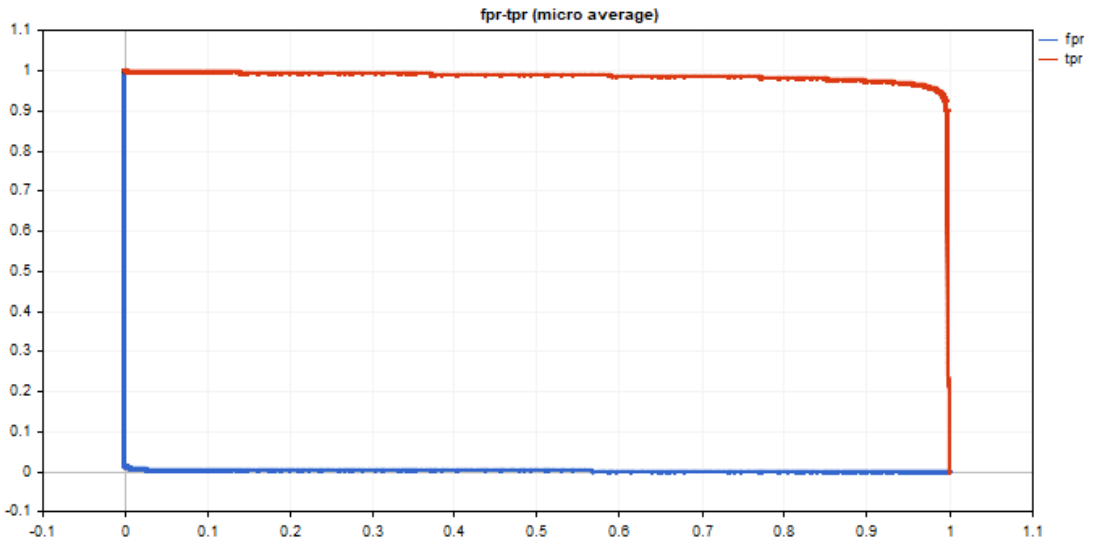
An example of plotting ROC graphs, where tpr values are plotted on the y-axis and fpr values are plotted on the x-axis. Also fpr and tpr graphs are plotted separately, with threshold values plotted on the x-axis

```
   matrixf mat_thres;
   matrixf mat_fpr;
   matrixf mat_tpr;
```

```
   if(y_true.ReceiverOperatingCharacteristic(y_scores,AVERAGE_MICRO,mat_fpr,mat_tpr,ma
     {
      double fpr[],tpr[],thres[];
      ArrayResize(fpr,mat_thres.Cols());
      ArrayResize(tpr,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<fpr.Size(); i++)
        {
         fpr[i]=mat_fpr[0][i];
         tpr[i]=mat_tpr[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("ROC curve (micro average)","roc","0.5",fpr,tpr);
      Plot2Curves("fpr-tpr (micro average)","fpr","tpr",thres,fpr,tpr);
     }
```

Resulting curves:





The graph output code is simple and based on the <Graphics/Graphic.mqh> standard library.

The examples use the data of the mnist.onnx model. The code is presented in the PrecisionRecall method description.

ROC AUC is close to ideal.

```
roc auc score micro = [0.99991]
```

Funzioni di Conversione

Questo è un gruppo di funzioni che forniscono la conversione dei dati da un formato ad un altro.

La funzione NormalizeDouble() dev' essere appositamente notata in quanto fornisce la necessaria precisione della presentazione del prezzo. Nelle operazioni di trading, in alcun modo i prezzi non normalizzati possono essere utilizzati se la loro precisione, anche di una cifra eccede quella richiesta dal tal trade server.

Funzione	Azione
CharToString	Conversione di un codice di simboli in una stringa di un carattere
DoubleToString	Conversione di un valore numerico ad una riga di testo con una precisione specificata
EnumToString	Conversione di un valore di enumerazione di qualsiasi tipo a stringa
NormalizeDouble	Arrotondamento di un numero in virgola mobile a una precisione specificata
StringToDouble	La conversione di una stringa contenente una rappresentazione simbolo di numero in numero di tipo double
StringToInteger	Conversione di una stringa contenente una rappresentazione di un simbolo di numero in numero di tipo long
StringToTime	Conversione di una stringa contenente l'ora o la data in formato di tipo datetime "aaaa.mm.gg [oo:mi]"
TimeToString	Conversione di un valore contenente il tempo in secondi trascorsi dal 01.01.1970 in una stringa di formato "aaaa.mm.gg. oo:mi" formato
IntegerToString	Conversione di int in una stringa di lunghezza predefinita
ShortToString	Conversione codice simbolo (unicode) in stringa costituida da un-simbolo
ShortArrayToString	Copia parte array in una stringa
StringToShortArray	Copia in senso-simbolo di una una stringa ad una parte dell' array selezionato di tipo ushort
CharArrayToString	Conversione del codice simbolo (ansi) in un array costituito da un-simbolo
StringToCharArray	Copia senso-simbolo di una string convertita da Unicode ad ANSI, in un luogo selezionato di array di tipo uchar
CharArrayToStruct	Copia l'array di tipo uchar sulla struttura POD
StructToCharArray	Copia struttura POD nell'array di tipo uchar
ColorToARGB	Conversione del tipo colore in tipo uint per ricevere la rappresentazione ARGB del colore.

Funzione	Azione
ColorToString	Conversione del valore del colore in stringa come "R,G,B"
StringToColor	Conversione stringa "R,G,B" o una stringa con il nome del colore in valore di tipo di colore
StringFormat	Conversione numero in stringa secondo il formato predefinito

Vedi anche

L'uso di una tabella Codici

CharToString

Conversione di un codice di simboli in una stringa di un-carattere.

```
string  CharToString(
   uchar  char_code      // codice numerico del simbolo
   );
```

Parametri

char_code

[in] Codice di simbolo ANSI.

Valore restituito

Stringa con un simbolo ANSI.

Vedi anche

StringToCharArray, ShortToString, StringGetCharacter

CharArrayToString

Esso copia e converte parte dell' array di tipo uchar in una stringa restituita.

```
string  CharArrayToString(
   uchar  array[],              // array
   int    start=0,              // posizione d'inizio dell'array
   int    count=-1              // numero di simboli
   uint    codepage=CP_ACP      // codice pagina
   );
```

Parametri

array[]

[In] Array di tipo uchar.

start=0

[in] Posizione da cui si avvia la copia. Per impostazione predefinita, viene utilizzato 0.

count=-1

[in] Numero di elementi di array per la copia. Definisce la lunghezza di una stringa risultante. Il valore predefinito è -1, il che significa la copia fino alla fine dell'array, o fino a terminale 0.

codepage=CP_ACP

[in] Il valore del codice pagina. Per i codici pagina più usati, provvedere appropriate costanti.

Valore restituito

String.

Vedi anche

StringToCharArray, ShortArrayToString, L'uso di una tabella Codici

CharArrayToStruct

Copia l'array di tipo uchar sulla struttura POD.

```
bool  CharArrayToStruct(
   void&          struct_object,     // struttura
   const uchar&   char_array[],      // array
   uint           start_pos=0        // posizione iniziale dell'array
   );
```

Parametri

struct_object

[in] Riferimento a qualsiasi tipo di struttura POD (contenente solo tipi di dati semplici).

char_array[]

[in] array di tipo uchar.

start_pos=0

[in] Posizione nell'array da cui inizia la copia dei dati.

Valore di ritorno

Returns true if successful, otherwise false.

Vedi anche

StringToCharArray, ShortArrayToString, StructToCharArray, Uso di Codepage, FileReadStruct, Unions (union), MathSwap

StructToCharArray

Copia struttura POD nell'array di tipo uchar.

```
bool  StructToCharArray(
   const void&  struct_object,      // struttura
   uchar&       char_array[],       // array
   uint         start_pos=0         // posizione iniziale nell'array
   );
```

Parametri

struct_object

[in] Riferimento a qualsiasi tipo di Struttura POD (contenente solo tipi di dati semplici).

char_array[]

[in] array di tipo uchar.

start_pos=0

[in] Posizione nell'array a partire dalla quale vengono aggiunti i dati copiati.

Valore di ritorno

Returns true if successful, otherwise false.

Note

Durante la copia, l'array dinamico si espande automaticamente (ArrayResize) se non c'è abbastanza spazio. Se l'array non può essere espanso fino al valore richiesto, la funzione restituisce un errore.

Vedi anche

StringToCharArray, ShortArrayToString,CharArrayToStruct, Uso di Codepage, FileWriteStruct, Unions (union), MathSwap

ColorToARGB

La funzione converte il tipo colore in tipo uint per ottenere rappresentazione ARGB del colore. Formato di colore ARGB viene utilizzato per generare una risorsa grafica, visualizzazione del testo, così come pure la classe libreria standard CCanvas.

```
uint  ColorToARGB(
   color  clr,          // colore convertito in formato color
   uchar  alpha=255     // canale alfa che gestisce la trasparenza del colore
   );
```

Parametri

clr

[in] Il valore del colore in variabile tipo di colore.

alpha

[in] Il valore del canale alfa usato per ricevere il colore nel formato ARGB. Il valore può essere impostato da 0 (un colore di un pixel di primo piano non cambia la visualizzazione di uno sottostante) fino a 255 (un colore di un pixel sottostante viene completamente sostituito dal pixel di primo piano). La trasparenza del colore in termini percentuali è calcolata come (1-alpha/255) * 100%. In altre parole, il valore minore del canale alfa porta ad un colore più trasparente.

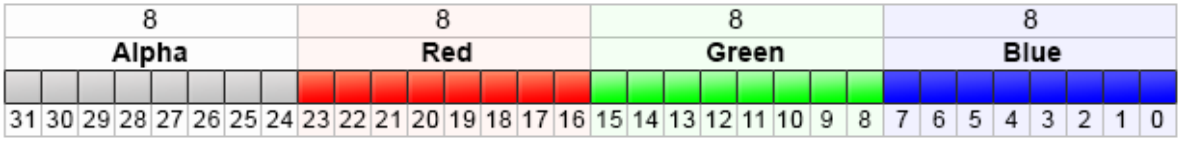
Valore restituito

Presentando il colore in formato ARGB dove Alfa, Rosso, Verde, Blu valori (canale alfa, rosso, verde, blu) sono impostati in serie, in quattro byte di tipo uint.

Nota

RGB è un formato di base e comunemente usato per la descrizione del colore del pixel su uno schermo in computer grafica. I nomi dei colori di base sono utilizzati per impostare componenti di colore rosso, verde e blu. Ogni componente è descritto da un byte specificando la saturazione di colore nell'intervallo da 0 a 255 (0x00 a 0xFF in formato esadecimale). Poiché il colore bianco contiene tutti i colori, è descritto come 0xFFFFFF, dove, ognuno di questi tre componenti viene presentato dal valore massimo di 0xFF.

Tuttavia, alcune operazioni richiedono di specificare la trasparenza del colore per descrivere l'aspetto di un'immagine nel caso esso sia coperto dal colore con un certo grado di trasparenza. Il concetto di canale alfa viene introdotto per tali casi. È implementato come un componente aggiuntivo di formato RGB. La struttura di formato ARGB è riportata di seguito.



Valori ARGB sono tipicamente espressi utilizzando formato esadecimale con ogni coppia di cifre che rappresentano i valori Alfa dei canali, rosso, verde e blu, i canali, rispettivamente. Ad esempio, il colore 80FFFF00 rappresenta il 50,2% giallo opaco. Inizialmente, 0x80 imposta valore alfa a 50,2%, come è 50,2% del valore 0xFF. Poi, la prima coppia FF definisce il valore massimo del componente rosso; la prossima coppia FF è simile alla precedente, ma per la componente verde; la coppia finale 00 rappresenta il valore più basso che il componente blu può avere (assenza di blu). La

combinazione di colori verde e rosso produce giallo. Se non si utilizza il canale alfa, la voce può essere ridotta fino al 6 cifre RRVVBB, questo è il motivo per cui i valori del canale alfa vengono memorizzati nei migliori bits di tipo uint.

A seconda del contesto, cifre esadecimali possono essere scritte con il prefisso '0x' o '#', per esempio, 80FFFF00, 0x80FFFF00 o #80FFFF00.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- imposta trasparenza
   uchar alpha=0x55;  // 0x55 significa 55/255=21.6 % di trasparenza
   //--- deriva la conversione ad ARGB per il colore clrBlue
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB con alpha=0x55 (trasparenza 21.6%%)",ColorToARGB
   //--- deriva la conversione in ARGB per il colore clrGreen
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB con alpha=0x55 (trasparenza 21.6%%)",ColorToARG
   //--- deriva la conversione in ARGB per il colore clrRed
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB con alpha=0x55 (trasparenza 21.6%%)",ColorToARGB
  }
```

Vedi anche

Resources, ResourceCreate(), TextOut(), color type, char, short, int and long types

ColorToString

Converte il valore del colore in stringa di forma "R,G,B".

```
string  ColorToString(
   color  color_value,     // valore del clore
   bool   color_name       // mostra o no il nome del colore
   );
```

Parametri

color_value

[in] Il valore del colore in variabile tipo di colore.

color_name

[in] Segno della necessità di restituire il nome del colore, se il nome del colore è identico a una delle costanti colore predefinite.

Valore restituito

Presentazione stringa del colore come "R,G,B", dove R,G e B sono costanti decimali tra 0 e 255 convertite in una stringa. Se il parametro color_name = true è impostato, si cercherà di convertire il valore di colore in nome di un colore.

Esempio:

```
   string clr=ColorToString(C'0,255,0'); // colore verde
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // ottiene la costante del colore
   Print(clr);
```

Vedi anche

StringToColor, ColorToARGB

DoubleToString

Conversione valore numerico nella stringa di testo.

```
string  DoubleToString(
   double  value,      // numero
   int     digits=8    // numero di cifre dopo il punto decimale
   );
```

Parametri

valore

[in] Valore di un floating point.

digits

[in] Formato accuratezza. Se il valore delle *cifre* è compreso tra 0 e 16, si otterrà una presentazione di stringa di un numero con il numero specificato di cifre dopo il punto. Se il valore delle *cifre* è compreso tra -1 e -16, si otterrà una rappresentazione di stringa di un numero nel formato scientifico con il numero specificato di cifre dopo il punto decimale. In tutti gli altri casi il valore stringa contiene 8 cifre dopo la virgola decimale.

Valore restituito

Stringa contenente una rappresentazione simbolo di un numero con la precisione specificata.

Esempio:

```
   Print("DoubleToString(120.0 + M_PI) : ",DoubleToString(120.0+M_PI));
   Print("DoubleToString(120.0 + M_PI,16) : ",DoubleToString(120.0+M_PI,16));
   Print("DoubleToString(120.0 + M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));
   Print("DoubleToString(120.0 + M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));
   Print("DoubleToString(120.0 + M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

Vedi anche

NormalizeDouble, StringToDouble

EnumToString

Conversione di un valore di enumerazione di qualsiasi tipo, in formato testo.

```
string  EnumToString(
   any_enum  value      // qualsiasi tipo di valore enumerazione
   );
```

Parametri

valore

[In] Qualsiasi tipo di valore di enumerazione.

Valore restituito

Una stringa con una rappresentazione testuale dell'enumerazione. Per ottenere il messaggio di errore chiamare la funzione GetLastError().

Nota

La funzione può impostare i seguenti valori di errore nella variabile _LastError:

- ERR_INTERNAL_ERROR - errore dell'ambiente di esecuzione
- ERR_NOT_ENOUGH_MEMORY - memoria non sufficiente per completare l'operazione
- ERR_INVALID_PARAMETER - non posso permettere il nome del valore dell'enumerazione

Esempio:

```
enum interval  // enumerazione di costanti nominate
  {
   month=1,     // intervallo un-mese
   two_months,  // due mesi
   quarter,     // tre mesi - un quarto
   halfyear=6,  // metà anno
   year=12,     // un anno - 12 mesi
  };
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- Imposta l'intervallo di tempo pari ad un mese
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- imposta l'intervallo di tempo equivalente ad un quarto (tre mesi)
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- imposta l'intervallo di tempo uguale ad un anno (12 mesi))
   period=year;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- controlla come viele mostrato il tipo di ordine
```

```
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- controlla come i valori scorretti vengono visualizzati
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// Risultato:
// mese=1
// quarto=3
// anno=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

Vedi anche

Enumerazioni, Variabili di input

IntegerToString

Questa funzione converte i valori di tipo integer in una stringa di lunghezza specificata e restituisce la stringa ottenuta.

```
string  IntegerToString(
   long    number,              // numero
   int     str_len=0,           // lunghezza della stringa risultante
   ushort  fill_symbol=' '      // filler
   );
```

Parametri

number

[in] Numero per la conversione.

str_len=0

[in] Lunghezza della stringa. Se la lunghezza della stringa risultante è maggiore di quella specificata, la stringa non viene tagliata. Se è inferiore, simboli riempitivi vengono aggiunti a sinistra.

fill_symbol=' '

[in] Filler symbol. Per impostazione predefinita è uno spazio.

Valore restituito

String.

Vedi anche

StringToInteger

ShortToString

Converte il codice del simbolo (unicode) in stringa di un-simbolo e restituisce la stringa risultante.

```
string  ShortToString(
   ushort  symbol_code      // simbolo
   );
```

Parametri

symbol_code

[in] Codice simbolo. Invece del codice simbolo è possibile utilizzare la stringa letterale che contiene un simbolo, o una stringa letterale con 2 byte di codice esadecimale corrispondente al simbolo dalla tabella Unicode.

Valore restituito

String.

Vedi anche

StringToCharArray, CharToString, StringGetCharacter

ShortArrayToString

Esso copia parte dell' array in una stringa restituita.

```
string  ShortArrayToString(
   ushort  array[],      // array
   int     start=0,      // posizione iniziale nell'array
   int     count=-1      // numero di simboli
   );
```

Parametri

array[]

[in] Array di tipo ushort (analogo per wchar_t type).

start=0

[in] Posizione, dal quale comincia la copia, Default - 0.

count=-1

[in] Numero di elementi dell' array da copiare. Definisce la lunghezza di una stringa risultante. Il valore predefinito è -1, il che significa la copia fino alla fine dell'array, o fino a terminale 0.

Valore restituito

String.

Vedi anche

StringToShortArray, CharArrayToString, L'uso di una tabella Codici

TimeToString

Conversione di un valore contenente tempo in secondi trascorsi dal 01.01.1970 in una stringa di formato "aaaa.mm.gg hh:mi".

```
string  TimeToString(
   datetime  value,                          // numero
   int       mode=TIME_DATE|TIME_MINUTES     // formato di output
   );
```

Parametri

valore

[in] Tempo in secondi da 00:00 del 1970/01/01.

mode=TIME_DATE|TIME_MINUTES

[in] Modalità di input dati addizionale. Può essere una flag o flag combinate:
TIME_DATE ottiene come risultato "aaaa.mm.gg",
TIME_MINUTES ottiene risultato come "hh:mi",
TIME_SECONDS ottiene risultati come "hh: mi: ss".

Valore restituito

String.

Vedi anche

StringToTime, TimeToStruct

NormalizeDouble

Arrotonda il numero floating point ad una precisione specificata.

```
double  NormalizeDouble(
   double   value,      // numero normalizzato
   int      digits      // numero di cifre dopo il punto decimale
   );
```

Parametri

valore

[in] Valore con il floating point.

digits

[in] Formato di accuratezza, numero di cifre dopo il punto (0-8).

Valore restituito

Valore di tipo double con precisione preset.

Nota

I valori calcolati di StopLoss, TakeProfit, e valori di prezzi di apertura da ordini pendenti, devono essere normalizzati con la precisione, il cui valore può essere ottenuto da Digits().

Si prega di notare che quando si emette in output nel Journal usando la funzione Print(), un numero normalizzato può contenere un maggior numero di cifre decimali che ci si aspetta. Ad esempio, per:

```
   double a=76.671;              // Un numero normalizzato con tre posti decimali
   Print("Print(76.671)=",a);    // Output come è
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // Output con un preset di accu
```

si avrà il seguente, nel terminale:

```
   DoubleToString(a,8)=76.67100000

   Print(76.671)=76.67100000000001
```

Esempio:

```
   double pi=M_PI;
   Print("pi = ",DoubleToString(pi,16));

   double pi_3=NormalizeDouble(M_PI,3);
   Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
   ;
   double pi_8=NormalizeDouble(M_PI,8);
   Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

   double pi_0=NormalizeDouble(M_PI,0);
   Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
/*
  Risultato:
```

```
   pi= 3.1415926535897931
   NormalizeDouble(pi,3)= 3.1419999999999999
   NormalizeDouble(pi,8)= 3.1415926499999998
   NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

Vedi anche

DoubleToString, Tipi Reali (double, float), Riduzione dei tipi

StringToCharArray

Copia in senso-simbolo una stringa convertita da Unicode ad ANSI, in un posto selezionato di un array di tipo uchar. Restituisce il numero di elementi copiati.

```
int  StringToCharArray(
   string  text_string,          // stringa sorgente
   uchar&  array[],              // array
   int     start=0,              // posizione iniziale nell'array
   int     count=-1              // numero di simboli
   uint    codepage=CP_ACP       // codice pagina
   );
```

Parametri

text_string

[in] Stringa da copiare .

array[]

[out] Array di tipo uchar.

start=0

[in] Posizione da cui si avvia la copia. Default - 0.

count=-1

[in] Numero di elementi dell' array da copiare. Definisce lunghezza di una stringa risultante. Il valore predefinito è -1, il che significa la copia fino alla fine dell'array, o fino a terminale 0. Terminale 0 verrà anche copiato nell' array destinatario, in questo caso la grandezza di un array dinamico può essere aumentata, se necessario, per la dimensione della stringa. Se la grandezza dell' array dinamico supera la lunghezza della stringa, la dimensione dell' array non sarà ridotta.

codepage=CP_ACP

[in] Il valore del codice pagina. Per i codici pagina più usati, provvedere appropriate costanti.

Valore restituito

Numero di elementi copiati.

Vedi anche

CharArrayToString, StringToShortArray, L'uso di una tabella Codici

StringToColor

Converte stringhe "R,G,B" o stringhe con il nome del colore in valore di tipo colore.

```
color  StringToColor(
   string  color_string      // rappresentazione stringa per colore
   );
```

Parametri

color_string

[in] Rappresentazione String di un colore di "R,G,B", o il nome di uno dei Web-colors predefiniti.

Valore restituito

Valore del colore.

Esempio:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- cambio il colore un po'
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

Vedi anche

ColorToString, ColorToARGB

StringToDouble

La funzione converte la stringa contenente una rappresentazione simbolo del numero in numero di tipo double.

```
double  StringToDouble(
   string  value      // stringa
   );
```

Parametri

valore

[in] Stringa contenente una rappresentazione simbolo di un numero.

Valore restituito

Valore di tipo double.

Vedi anche

NormalizeDouble, Tipi Reali (double, float), Riduzione dei tipi

StringToInteger

La funzione converte una stringa contenente una rappresentazione simbolo di numero in numero di tipo long (intero).

```
long  StringToInteger(
   string  value      // stringa
   );
```

Parametri

valore

[in] Stringa contenente un numero.

Valore restituito

Valore di tipo long.

Vedi anche

IntegerToString, Tipi Reali (double, float), Riduzione dei tipi

StringToShortArray

La funzione copia in senso-simbolo una stringa in un posto specificato di un array di tipo ushort. Restituisce il numero di elementi copiati.

```
int  StringToShortArray(
   string  text_string,      // stringa sorgente
   ushort& array[],          // array
   int     start=0,          // posizione iniziale nell'array
   int     count=-1          // numero di simboli
   );
```

Parametri

text_string

[in] Stringa da copiare

array[]

[out] Array of ushort type (analog of wchar_t type).

start=0

[in] Posizione, da cui ha inizio la copia. Default - 0.

count=-1

[in] Numero di elementi dell' array da copiare. Definisce lunghezza di una stringa risultante. Il valore predefinito è -1, che significa copiare fino alla fine dell' array o sino terminale 0. Terminal 0 sarà anche copiato nell'array destinazione, in questo caso la grandezza dell' array dinamico può essere aumentata se necessario alla grandezza della stringa. Se la grandezza dell' array dinamico supera la lunghezza della stringa, la dimensione dell' array non sarà ridotta.

Valore restituito

Numero di elementi copiati.

Vedi anche

ShortArrayToString, StringToCharArray, L'uso di una tabella Codici

StringToTime

Trasforma la stringa contenente il tempo e/o la data nel formato "aaaa.mm.gg [oo:mi]" in numero di tipo datetime.

```
datetime  StringToTime(
   const string  time_string      // data della stringa
   );
```

Parametri

time_string

[in] Stringa in uno dei formati specificati:

- "aaaa.mm.gg [oo:mi]"
- "aaaa.mm.gg [oo:mi:ss]"
- "aaaammgg [oo:mi:ss]"
- "aaaammgg [oomiss]"
- "aaaa/mm/gg [oo:mi:ss]"
- "aaaa-mm-gg [oo:mi:ss]"

Valore di ritorno

Valore di tipo datetime contenente il numero di secondi trascorsi dal 01.01.1970.

Nota

Qualsiasi sequenza di spazi e caratteri di tabulazione tra data ed ora è considerata come uno spazio singolo per evitare un'ulteriore elaborazione di *time_string* prima di chiamare StringToTime().

Guarda anche

TimeToString, TimeToStruct

StringFormat

I parametri ottenuti dalle funzioni formato e restituzione di stringa

```
string  StringFormat(
   string  format,     // stringa con la descrizione del formato
   ...     ...         // parametri
   );
```

Parametri

(formato)

[in] Stringa contenente i metodi di formattazione. Le regole di formattazione sono le stesse per la funzione PrintFormat.

...

[in] Parametri, separati da virgola.

Valore restituito

String.

Esempio:

```
void OnStart()
  {
//--- string variables
   string output_string;
   string temp_string;
   string format_string;
//--- prepare the specification header
   temp_string=StringFormat("Contract specification for %s:\n",_Symbol);
   StringAdd(output_string,temp_string);
//--- int value output
   int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   temp_string=StringFormat("   SYMBOL_DIGITS = %d (number of digits after the decimal
                            digits);
   StringAdd(output_string,temp_string);
//--- double value output with variable number of digits after the decimal point
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   format_string=StringFormat("   SYMBOL_POINT = %%.%df (point value)\n",
                              digits);
   temp_string=StringFormat(format_string,point_value);
   StringAdd(output_string,temp_string);
//--- int value output
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   temp_string=StringFormat("   SYMBOL_SPREAD = %d (current spread in points)\n",
                            spread);
   StringAdd(output_string,temp_string);
//--- int value output
   int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_STOPS_LEVEL = %d (minimal indention in po
                            min_stop);
   StringAdd(output_string,temp_string);
//--- double value output without the fractional part
   double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_CONTRACT_SIZE = %.f (contract size)\n",
                            contract_size);
   StringAdd(output_string,temp_string);
//--- double value output with default accuracy
   double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_TICK_SIZE = %f (minimal price change)\n",
                            tick_size);
   StringAdd(output_string,temp_string);
//--- determining the swap calculation mode
   int swap_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_SWAP_MODE);
   string str_swap_mode;
   switch(swap_mode)
     {
      case SYMBOL_SWAP_MODE_DISABLED: str_swap_mode="SYMBOL_SWAP_MODE_DISABLED (no swa
      case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (in points)
      case SYMBOL_SWAP_MODE_CURRENCY_SYMBOL: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_MARGIN: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY
      case SYMBOL_SWAP_MODE_INTEREST_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST
      case SYMBOL_SWAP_MODE_INTEREST_OPEN: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_OP
      case SYMBOL_SWAP_MODE_REOPEN_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_CUR
      case SYMBOL_SWAP_MODE_REOPEN_BID: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_BID (by
     }
//--- string value output
   temp_string=StringFormat("   SYMBOL_SWAP_MODE = %s\n",
                            str_swap_mode);
   StringAdd(output_string,temp_string);
//--- double value output with default accuracy
   double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
```

```
   temp_string=StringFormat("   SYMBOL_SWAP_LONG = %f (long swap value)\n",
                            swap_long);
   StringAdd(output_string,temp_string);
//--- double value output with default accuracy
   double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
   temp_string=StringFormat("   SYMBOL_SWAP_SHORT = %f (short swap value)\n",
                            swap_short);
   StringAdd(output_string,temp_string);
//--- determining the trading mode
   int trade_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_MODE);
   string str_trade_mode;
   switch(trade_mode)
     {
      case SYMBOL_TRADE_MODE_DISABLED: str_trade_mode="SYMBOL_TRADE_MODE_DISABLED (tra
      case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY (onl
      case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY (o
      case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY (o
      case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (no trade re
     }
//--- string value output
   temp_string=StringFormat("   SYMBOL_TRADE_MODE = %s\n",
                            str_trade_mode);
   StringAdd(output_string,temp_string);
//--- double value output in a compact format
   double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   temp_string=StringFormat("   SYMBOL_VOLUME_MIN = %g (minimal volume for a deal)\n",
   StringAdd(output_string,temp_string);
//--- double value output in a compact format
   double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
   temp_string=StringFormat("   SYMBOL_VOLUME_STEP = %g (minimal volume change step)\n
   StringAdd(output_string,temp_string);
//--- double value output in a compact format
   double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
   temp_string=StringFormat("   SYMBOL_VOLUME_MAX = %g (maximal volume for a deal)\n",
   StringAdd(output_string,temp_string);
//--- determining the contract price calculation mode
   int calc_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_CALC_MODE);
   string str_calc_mode;
   switch(calc_mode)
     {
      case SYMBOL_CALC_MODE_FOREX:str_calc_mode="SYMBOL_CALC_MODE_FOREX (Forex)";breal
      case SYMBOL_CALC_MODE_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_FUTURES (futures)"
      case SYMBOL_CALC_MODE_CFD:str_calc_mode="SYMBOL_CALC_MODE_CFD (CFD)";break;
      case SYMBOL_CALC_MODE_CFDINDEX:str_calc_mode="SYMBOL_CALC_MODE_CFDINDEX (CFD for
      case SYMBOL_CALC_MODE_CFDLEVERAGE:str_calc_mode="SYMBOL_CALC_MODE_CFDLEVERAGE (C
      case SYMBOL_CALC_MODE_EXCH_STOCKS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_STOCKS (t
      case SYMBOL_CALC_MODE_EXCH_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES
      case SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FU
     }
//--- string value output
   temp_string=StringFormat("   SYMBOL_TRADE_CALC_MODE = %s\n",
                            str_calc_mode);
   StringAdd(output_string,temp_string);
//--- double value output with 2 digits after the decimal point
   double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
   temp_string=StringFormat("   SYMBOL_MARGIN_INITIAL = %.2f (initial margin)\n",
                            margin_initial);
   StringAdd(output_string,temp_string);
//--- double value output with 2 digits after the decimal point
   double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
   temp_string=StringFormat("   SYMBOL_MARGIN_MAINTENANCE = %.2f (maintenance margin)\
```

```
                              margin_maintenance);
   StringAdd(output_string,temp_string);
//--- int value output
   int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_FREEZE_LEVEL = %d (order freeze level in
                              freeze_level);
   StringAdd(output_string,temp_string);
   Print(output_string);
   Comment(output_string);
/* execution result
   Contract specification for EURUSD:
     SYMBOL_DIGITS = 5 (number of digits after the decimal point)
     SYMBOL_POINT = 0.00001 (point value)
     SYMBOL_SPREAD = 10 (current spread in points)
     SYMBOL_TRADE_STOPS_LEVEL = 18 (minimal indention in points for Stop orders)
     SYMBOL_TRADE_CONTRACT_SIZE = 100000 (contract size)
     SYMBOL_TRADE_TICK_SIZE = 0.000010 (minimal price change)
     SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (in points)
     SYMBOL_SWAP_LONG = -0.700000 (buy order swap value)
     SYMBOL_SWAP_SHORT = -1.000000 (sell order swap value)
     SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (no trade restrictions)
     SYMBOL_VOLUME_MIN = 0.01 (minimal volume for a deal)
     SYMBOL_VOLUME_STEP = 0.01 (minimal volume change step)
     SYMBOL_VOLUME_MAX = 500 (maximal volume for a deal)
     SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
     SYMBOL_MARGIN_INITIAL = 0.00 (initial margin)
     SYMBOL_MARGIN_MAINTENANCE = 0.00 (maintenance margin)
     SYMBOL_TRADE_FREEZE_LEVEL = 0 (order freeze level in points)
*/
  }
```

Vedi anche

PrintFormat, DoubleToString,ColorToString, TimeToString

Funzioni di matematica

Un insieme di funzioni matematiche e trigonometriche.

Math functions were originally designed to perform relevant operations on scalar values. From this build on, most of the functions can be applied to matrices and vectors. These include MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, and MathTanh. Such operations imply element-wise handling of matrices or vectors. Example:

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

For MathMod and MathPow, the second element can be either a scalar or a matrix/vector of the appropriate size.

Funzione	Azione
MathAbs	Restituisce il valore assoluto (modulo) del valore numerico specificato
MathArccos	Restituisce l'arcocoseno di x in radianti
MathArcsin	Restituisce l'arcoseno di x in radianti
MathArctan	Restituisce l'arcotangente di x in radianti
MathArctan2	Restituisce l'arcotangente del quoziente di due argomenti (x, y)
MathCeil	Restituisce il valore numerico integer(intero) più vicino dall'alto
MathCos	Restituisce il coseno di un numero
MathExp	Restituisce l'esponente di un numero
MathFloor	Restituisce il valore numerico integer(intero) più vicino dal basso
MathLog	Restituisce il logaritmo naturale
MathLog10	Restituisce il logaritmo di un numero da base 10
MathMax	Restituisce il valore massimo dei due valori numerici

Funzione	Azione
MathMin	Restituisce il valore minimo dei due valori numerici
MathMod	Restituisce il resto vero dopo la divisione di due numeri
MathPow	Aumenta la base alla potenza specificata
MathRand	Restituisce un valore pseudocasuale nell'intervallo da 0 a 32767
MathRound	Arrotonda un valore al numero intero più vicino
MathSin	Restituisce il seno di un numero
MathSqrt	Restituisce una radice quadrata
MathSrand	Consente di impostare il punto di partenza per la generazione di una serie di interi pseudocasuali
MathTan	Restituisce la tangente di un numero
MathIsValidNumber	Verifica la correttezza di un numero reale
MathExpm1	Returns the value of the expression MathExp(x)-1
MathLog1p	Returns the value of the expression MathLog(1+x)
MathArccosh	Returns the hyperbolic arccosine
MathArcsinh	Returns the hyperbolic arcsine
MathArctanh	Returns the hyperbolic arctangent
MathCosh	Returns the hyperbolic cosine
MathSinh	Returns the hyperbolic sine
MathTanh	Returns the hyperbolic tangent
MathSwap	Cambia l'ordine dei byte nei tipi valori ushort/uint/ushort

MathAbs

La funzione restituisce il valore assoluto (modulo) di valore numerico specificato.

```
double  MathAbs(
   double  value      // valore numerico
   );
```

Parametri

valore

[in] Numeric value.

Valore restituito

Valore di tipo double maggiore o uguale a zero.

Nota

Al posto della funzione MathAbs() è possibile utilizzare fabs().

MathArccos

La funzione restituisce l'arcocoseno di x nel campo da 0 a π in radianti.

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

Parametri

val

[A] Il valore val tra -1 e 1, il quale arcocoseno deve essere calcolato.

Valore restituito

Arcocoseno di un numero in radianti. Se val è minore di -1 o maggiore di 1, la funzione restituisce NaN (valore indeterminato).

Nota

Al posto della funzione MathArccos() è possibile utilizzare acos().

Vedi anche

I tipi reali (double, float)

MathArcsin

La funzione restituisce l'arcoseno di x entro il range di -π/2 a π/2 radianti.

```
double  MathArcsin(
   double  val      // -1<valore<1
   );
```

Parametri

val

[A] Il valore val tra -1 e 1, l'arcoseno che deve essere calcolato.

Valore restituito

Arcoseno del numero val in radianti nell'intervallo da -π/2 a π/2 radianti. Se val è minore di -1 o maggiore di 1, la funzione restituisce NaN (valore indeterminato).

Nota

Al posto della funzione MathArcsin() è possibile utilizzare asin().

Vedi anche

I tipi reali (double, float)

MathArctan

La funzione restituisce l'arcotangente di x. Se x è uguale a 0, la funzione restituisce 0.

```
double  MathArctan(
   double  value      // tangente
   );
```

Parametri

valore

[in] Un numero che rappresenta la tangente.

Valore restituito

MathArctan restituisce un valore compreso tra -π/2 e π/2 radianti.

Nota

Al posto della funzione MathArctan() è possibile utilizzare atan().

MathArctan2

Restituisce l'arcotangente del quoziente di due argomenti (x, y).

```
double  MathArctan2(
   const double       y,          // Coordinate Y
   const double       x           // Coordinate X
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Parametri

y

[in] Le coordinate Y del punto.

x

[in] Le coordinate X del punto.

Note

Si prega di notare quanto segue.

- Per (x, y) nel quadrante 1, il valore restituito sarà: 0 < θ < π/2.
- Per (x, y) nel quadrante 2, il valore restituito sarà: π/2 <θ≤π.
- Per (x, y) nel quadrante 3, il valore restituito sarà: -π <θ <-π/2.
- Per (x, y) nel quadrante 4, il valore restituito sarà: -π/2 <θ <0.

Il valore di ritorno per i punti fuori questi quadranti è indicato di seguito.

- Se y è 0 e x non è negativo, allora θ = 0.
- Se y è 0 e x è negativo, allora θ = π.
- Se y è un numero positivo, e x è 0, allora θ = π/2.
- Se y è negativo e x è 0, allora θ = -π/2.
- Se y è 0 e x è 0, allora θ = -π/2.

Se il valore del parametro X o Y è NaN, o se i valori dei parametri x ed y sono uguali al valore PositiveInfinity o NegativeInfinity, il metodo restituisce il valore NaN.

MathClassify

Determines the type of a real number and returns a result as a value from the ENUM_FP_CLASS enumeration

```
ENUM_FP_CLASS  MathClassify(
   double  value      // real number
   );
```

Parameters

value

[in] The real number to be checked

Return Value

A value from the ENUM_FP_CLASS enumeration

ENUM_FP_CLASS

ID	Description
FP_SUBNORMAL	A subnormal number which is closer to zero than the smallest representable normal number DBL_MIN (2.2250738585072014e-308)
FP_NORMAL	A normal number in the range between 2.2250738585072014e-308 and 1.7976931348623158e+308
FP_ZERO	A positive or a negative zero
FP_INFINITE	A number which cannot be represented by the appropriate type, positive or negative infinity
FP_NAN	Not a number.

Esempio:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- test NaN
  double nan=double("nan");
  PrintFormat("Test NaN: %G is %s, MathIsValidNumber(NaN)=%s",
              nan,
              EnumToString(MathClassify(nan)),
              (string)MathIsValidNumber(nan));
//--- test infinity
  double inf=double("inf");
  PrintFormat("Test Inf: %G is %s, MathIsValidNumber(inf)=%s",
```

```
               inf,
               EnumToString(MathClassify(inf)),
               (string)MathIsValidNumber(inf));
//--- test normal value
  double normal=1.2345e6;
  PrintFormat("Test Normal: %G is %s, MathIsValidNumber(normal)=%s",
               normal,
               EnumToString(MathClassify(normal)),
               (string)MathIsValidNumber(normal));
//--- test subnormal value
  double sub_normal=DBL_MIN/2.0;
  PrintFormat("Test Subnormal: %G is %s, MathIsValidNumber(sub_normal)=%s",
               sub_normal,
               EnumToString(MathClassify(sub_normal)),
               (string)MathIsValidNumber(sub_normal));
//--- test zero value
  double zero=0.0/(-1);
  PrintFormat("Test Zero: %G is %s, MathIsValidNumber(zero)=%s",
               zero,
               EnumToString(MathClassify(zero)),
               (string)MathIsValidNumber(zero));
 }
 /*
 Result:
   Test NaN: NAN is FP_NAN, MathIsValidNumber(NaN)=false
   Test Inf: INF is FP_INFINITE, MathIsValidNumber(inf)=false
   Test Normal: 1.2345E+06 is FP_NORMAL, MathIsValidNumber(normal)=true
   Test Subnormal: 1.11254E-308 is FP_SUBNORMAL, MathIsValidNumber(sub_normal)=true
   Test Zero: -0 is FP_ZERO, MathIsValidNumber(zero)=true
*/
//+------------------------------------------------------------------+
```

See also

Real types (double, float), MathIsValidNumber

MathCeil

La funzione restituisce un valore numerico intero più vicino dall'alto.

```
double  MathCeil(
   double  val      // numero
   );
```

Parametri

val

[in] Numeric value.

Valore restituito

Valore numerico che rappresenta il più piccolo intero che supera o è uguale a val.

Nota

Al posto della funzione MathCeil() si può usare ceil().

MathCos

La funzione restituisce il coseno di un angolo.

```
double  MathCos(
   double  value      // numero
   );
```

Parametri

valore

[in] Angolo in radianti.

Valore restituito

Valore di tipo double compreso tra -1 e 1.

Nota

Invece di MathCos() è possibile utilizzare cos().

MathExp

La funzione restituisce il valore di e elevato alla potenza di d.

```
double  MathExp(
   double  value      // potenza per il numero e
   );
```

Parametri

valore

[in] Un numero che specifica la potenza.

Valore restituito

Un certo numero di tipo double. All' overflow, la funzione restituisce INF (infinito), in caso di perdita orgine MathExp restituisce 0.

Nota

Al posto di MathExp() è possibile utilizzare exp().

Vedi anche

I tipi reali (double, float)

MathFloor

La funzione restituisce un valore numerico intero più vicino dal basso.

```
double  MathFloor(
   double  val     // numero
   );
```

Parametri

val

[in] Numeric value.

Valore restituito

Un valore numerico che rappresenta il numero intero che è minore o uguale a val.

Nota

Invece di MathFloor() è possibile utilizzare floor().

MathLog

La funzione restituisce un logaritmo naturale.

```
double  MathLog(
   double  val      // valore da cui prendere il logaritmo
   );
```

Parametri

val

[in] Valore logaritmo in cui va trovato.

Valore restituito

Il logaritmo naturale di val in caso di successo. Se val è negativo, la funzione restituisce NaN (valore indeterminato). Se val è uguale a 0, la funzione restituisce INF (infinito).

Nota

Invece di MathLog() è possibile utilizzare log().

Vedi anche

I tipi reali (double, float)

MathLog

Restituisce il logaritmo di un numero da base 10.

```
double  MathLog10(
   double  val      // numero da cui prendere il logaritmo
   );
```

Parametri

val

[in] Valore numerico del logaritmo comune, che deve essere calcolato.

Valore restituito

Il logaritmo comune in caso di successo. Se val è negativo, la funzione restituisce NaN (valore indeterminato). Se val è uguale a 0, la funzione restituisce INF (infinito).

Nota

Invece di MathLog10() è possibile utilizzare log10().

Vedi anche

I tipi reali (double, float)

MathMax

La funzione restituisce il valore massimo di due valori.

```
double  MathMax(
   double  value1,      // primo valore
   double  value2       // secondo valore
   );
```

Parametri

value1

[in] Primo valore numerico.

value2

[in] Secondo valore numerico.

Valore restituito

Il più grande dei due valori.

Nota

Invece di Mathmax() è possibile utilizzare fmax(). Functions fmax(), fmin(), MathMax(), MathMin() può lavorare con tipi interi senza fare il loro typecasting per il tipo double.

Se i parametri di tipi diversi vengono passati in una funzione, il parametro del tipo minore è automaticamente castato(da "casting") al tipo principale. Il tipo del valore di ritorno corrisponde al tipo principale.

Se i dati dello stesso tipo sono passati, nessun casting viene eseguito.

MathMin

La funzione restituisce il valore minimo di due valori.

```
double  MathMin(
   double  value1,      // primo valore
   double  value2       // secondo valore
   );
```

Parametri

value1

[in] Primo valore numerico.

value2

[in] Secondo valore numerico.

Valore restituito

Il più piccolo dei due valori.

Nota

Invece di MathMin() è possibile utilizzare fmin(). Le funzioni fmax(), fmin(), MathMax(), MathMin() possono lavorare con tipi interi senza fare il typecasting di queste al tipo double.

Se i parametri di tipi diversi vengono passati in una funzione, il parametro del tipo minore è automaticamente castato(da "casting") al tipo principale. Il tipo del valore di ritorno corrisponde al tipo principale.

Se i dati dello stesso tipo sono passati, nessun casting viene eseguito.

MathMod

La funzione restituisce il resto della divisione reale di due numeri.

```
double  MathMod(
   double  value,      // valore dividendo
   double  value2      // valore divisore
   );
```

Parametri

valore

[in] Valore dividendo.

value2

[in] Valore Divisore.

Valore restituito

La funzione MathMod calcola il valore reale di f di val / y tale che val = i * y + f, dove i è un numero intero, f ha lo stesso segno val, ed il valore assoluto di f è minore del valore assoluto di y.

Nota

Al posto di MathMod() è possibile utilizzare fmod().

MathPow

La funzione eleva una base per una potenza specificata.

```
double  MathPow(
   double  base,          // base
   double  exponent       // valore dell'esponente
   );
```

Parametri

base

[in] Base.

esponente

[in] Valore esponente.

Valore restituito

Valore della base elevato alla potenza specificata.

Nota

Al posto di MathPow() è possibile utilizzare pow().

MathRand

Restituisce un numero intero pseudocasuale compreso tra 0 e 32767.

```
int  MathRand();
```

Valore restituito

Il valore intero compreso tra 0 e 32767.

Nota

Prima della prima chiamata della funzione, è necessario chiamare MathSrand per impostare il generatore di numeri pseudocasuali allo stato iniziale.

Nota

Invece di MathRand() è possibile utilizzare rand().

MathRound

La funzione restituisce un valore arrotondato al numero intero più vicino al valore numerico specificato.

```
double  MathRound(
   double  value      // valore da arrotondare
   );
```

Parametri

valore

[in] Valore numerico prima di arrotondare.

Valore restituito

Valore arrotondato fino al numero intero più vicino.

Nota

Invece di MathRound() è possibile utilizzare round().

MathSin

Restituisce il seno di un angolo specificato.

```
double  MathSin(
   double  value      // argomento in radianti
   );
```

Parametri

valore

[in] Angolo in radianti.

Valore restituito

Seno di un angolo misurato in radianti. Restituisce il valore compreso tra -1 e 1.

Nota

Invece di MathSin() è possibile utilizzare sin().

MathSqrt

Restituisce la radice quadrata di un numero.

```
double  MathSqrt(
   double  value      // numero positivo
   );
```

Parametri

valore

[in] Valore numerico positivo.

Valore restituito

Radice quadrata del valore. Se il valore è negativo, MathSqrt restituisce NaN (valore indeterminato).

Nota

Al posto di MathSqrt() è possibile utilizzare sqrt().

Vedi anche

I tipi reali (double, float)

MathSrand

Consente di impostare il punto di partenza per la generazione di una serie di interi pseudocasuali.

```
void  MathSrand(
   int  seed      // numero di inizializzazione
   );
```

Parametri

seme

[in] Il numero di partenza per la sequenza di numeri casuali.

Valore restituito

Nessun valore restituito.

Nota

La funzione MathRand() è usata per generare una sequenza di numeri pseudocasuali. La chiamata di MathSrand() con un certo numero di inizializzazione consente di produrre sempre la stessa sequenza di numeri pseudocasuali.

Per assicurare la ricezione di seuenze non-ricorrenti, utilizzare la chiamata di MathSrand(GetTickCount()), in quanto il valore di GetTickCount() aumenta dal momento di inizio del sistema operativo e non viene ripetuto entro 49 giorni, fino a quando il contatore di millisecondi incorporato, non overflowa. L' uso di MathSrand (TimeCurrent()) non è adatto, poiché TimeCurrent() restituisce il tempo dell'ultimo tick, che può essere invariato per lungo tempo, ad esempio durante il fine settimana.

Initialization of the random number generator using MathSrand() for indicators and Expert Advisors is better performed in the OnInit() handler; it saves you from the following multiple restarts of the generator in OnTick() and OnCalculate().

Al posto della funzione MathSrand() è possibile utilizzare la funzione srand().

Esempio:

```
#property description "L'indicatore mostra il teorema del limite centrale, che dice:"
#property description "La somma di un numero sufficientemente grande di variabili casu
#property description "avente approssimativamente uguale magnitudo (nessuno degli adde
#property description "o rende un contributo determinante alla somma), ha una distribu

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Proprietà della costruzione grafica
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- Una variabile di input
input int      sample_number=10;
```

```
//--- Un buffer indicatore per disegnare la distribuzione
double         LabelBuffer[];
//--- Un contatore di ticks
double         ticks_counter;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Associa un array e un buffer indicatore
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- porta il buffer indicatore vicino dal presente al passato
   ArraySetAsSeries(LabelBuffer,true);
//--- Inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- Inizializza il contatore di ticks
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- Per un contatore zero, resetta il buffer indicatore
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- Incrementa il contatore
   ticks_counter++;
//--- Dobbiamo resettare periodicamente i ticks contatore, per ravvivare la distribuz
   if(ticks_counter>100)
     {
      Print("Abbiamo resettato i valori dell'indicatore, iniziamo a riempire le celle
      ticks_counter=0;
     }
//--- Ottiene un campione di valori casuali come somma di tre numeri da 0 a 7
   for(int i=0;i<sample_number;i++)
     {
      //--- Calcola l'indice delle celle, dove il numero casuale cade tra la somma di
      int rand_index=0;
      //--- Ottiene tre numeri casuali da 0 a 7
      for(int k=0;k<3;k++)
```

```
         {
          //--- Il resto della divisione per 7 restituità un valore da 0 a 6
          rand_index+=MathRand()%7;
         }
       //--- Incrementa il valore nel numero della cella rand_index by 1
       LabelBuffer[rand_index]++;
      }
//--- Esce dall'handoler OnCalculate()
   return(rates_total);
  }
```

MathTan

La funzione restituisce la tangente di un numero.

```
double  MathTan(
   double  rad      // argomento in radianti
   );
```

Parametri

rad

[in] Angolo in radianti.

Valore restituito

Tangente di rad. Se rad è maggiore o uguale a 263, o inferiore o uguale a -263, una perdita di significatività del risultato si verifica, nel qual caso la funzione restituisce un numero indefinito.

Nota

Al posto di MathTan() è possibile utilizzare tan().

Vedi anche

I tipi reali (double, float)

MathIsValidNumber

Verifica la correttezza di un numero reale.

```
bool  MathIsValidNumber(
   double  number      // numero da verificare
   );
```

Parametri

number

[in] Valore numerico controllato.

Valore restituito

Restituisce true se il valore controllato è un numero accettabile reale. Se il valore selezionato è un infinito più o meno, o "non un numero" (NaN), la funzione restituisce il valore false.

Esempio:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Attenzione! MathArcsin(2.0) = ",abnormal);
```

Vedi anche

Real types (double, float)

MathExpm1

Returns the value of the expression MathExp(x)-1.

```
double  MathExpm1(
   double  value      // power for the number e
   );
```

Parameters

value

[in] The number specifying the power.

Return Value

A value of the double type. In case of overflow the function returns INF (infinity), in case of underflow MathExpm1 returns 0.

Note

At values of *x* close to 0, the MathExpm1(x) function generates much more accurate values than the MathExp(x)-1 function.

Instead of the MathExpm1() function you can use the expm1() function.

See also

Real types (double, float)

MathLog1p

Returns the value of the expression MathLog(1+x).

```
double  MathLog1p(
   double  value      // value to take the logarithm
   );
```

Parameters

value

[in] The value, the logarithm of which is to be calculated.

Return Value

The natural logarithm of the value (value + 1) if successful. If value is < -1, the function returns NaN (undefined value). If value is equal to -1, the function returns INF (infinity).

Note

At values of x close to 0, the MathLog1p(x) function generates much more accurate values than the MathLog(1+x) function.

Instead of the MathLog1p() function you can use the log1p() function.

See also

Real types (double, float)

MathArccosh

Returns the hyperbolic arccosine.

```
double  MathArccosh(
   double  value     // 1 <= value < ∞
   );
```

Parameters

value

[in] The value, the hyperbolic arccosine of which is to be calculated.

Return Value

The hyperbolic arccosine of the number. If value is less than +1, the function returns NaN (undefined value).

Note

Instead of the MathArccosh() function you can use the acosh() function.

See also

Real types (double, float)

MathArcsinh

Returns the hyperbolic arcsine.

```
double  MathArcsinh(
   double  value      // -∞ < value < +∞
   );
```

Parameters

val

[in] The value, the hyperbolic arcsine of which is to be calculated.

Return Value

The hyperbolic arcsine of the number.

Note

Instead of the MathArcsinh() function you can use the asinh() function.

See also

Real types (double, float)

MathArctanh

Returns the hyperbolic arctangent.

```
double  MathArctanh(
   double  value      // value in the range of -1 < value < 1
   );
```

Parameters

value

[in] Number within the range of -1 < value < 1, which represents the tangent.

Return Value

The hyperbolic arctangent of the number.

Note

Instead of the MathArctanh() function you can use the atanh() function.

MathCosh

Returns the hyperbolic cosine of the number.

```
double  MathCosh(
   double  value      // number
   );
```

Parameters

value

[in] Value.

Return Value

The hyperbolic cosine of the number, value within the range of +1 to positive infinity.

Note

Instead of the MathCosh() function you can use the cosh() function.

MathSinh

Returns the hyperbolic sine of the number.

```
double  MathSinh(
   double  value      // number
   );
```

Parameters

value

[in] Value.

Return Value

The hyperbolic sine of the number.

Note

Instead of the MathSinh() function you can use the sinh() function.

MathTanh

Returns the hyperbolic tangent of the number.

```
double  MathTanh(
   double  value      // number
   );
```

Parameters

value

[in] Value.

Return Value

The hyperbolic tangent of the number, value within the range of -1 to +1.

Note

Instead of the MathTanh() function you can use the tanh() function.

See also

Real types (double, float)

MathSwap

Cambia l'ordine dei byte nei valori di tipo ushort.

```
ushort  MathSwap(
   ushort  value      // valore
   );
```

Parametri

valore

[in] Valore per la modifica dell'ordine dei byte.

Valore di ritorno

valore ushort con l'ordine dei byte inverso.

MathSwap

Cambia l'ordine dei byte nel tipo di valore uint.

```
uint  MathSwap(
   uint   value      // valore
   );
```

Parametri

valore

[in] Valore per la modifica dell'ordine dei byte.

Valore di ritorno

valore uint con l'ordine dei byte inverso.

MathSwap

Cambia l'ordine dei byte nel tipo di valore ulong.

```
ulong  MathSwap(
   ulong  value      // valore
   );
```

Parametri

valore

[in] Valore per la modifica dell'ordine dei byte.

Valore di ritorno

valore ulong con l'ordine dei byte inverso.

Funzioni Stringa

Questo è un gruppo di funzioni destinate a lavorare con i dati del tipo stringa.

Funzione	Azione
StringAdd	Aggiunge una stringa alla fine di un'altra stringa
StringBufferLen	Restituisce la grandezza del buffer allocato per la stringa
StringCompare	Confronta due stringhe e restituisce 1 se la prima stringa è maggiore della seconda; 0 - se le stringhe sono uguali; -1 (meno 1) - se la prima stringa è minore della seconda
StringConcatenate	Forma una serie di parametri passati
StringFill	Riempie una stringa specificata da simboli selezionati
StringFind	Ricerca di una sottostringa in una stringa
StringGetCharacter	Restituisce il valore di un numero situato nella posizione specificata della stringa
StringInit	Inizializza la stringa per simboli specificati e fornisce la lunghezza della stringa specificata
StringLen	Restituisce il numero di simboli in una stringa
StringSetLength	Imposta una lunghezza specificata (in caratteri) per una stringa
StringReplace	Sostituisce tutte le sottostringhe trovate di una stringa da una sequenza di set di simboli
StringReserve	Riserva il buffer di una grandezza specificata per una stringa, in memoria
StringSetCharacter	Restituisce una copia di una stringa con un valore modificato di un simbolo in una posizione specificata
StringSplit	Ottiene sottostringhe da un separatore specificato dalla stringa specificata, restituisce il numero di sottostringhe ottenute
StringSubstr	Estrae una sottostringa da una stringa di testo a partire da una posizione specificata
StringToLower	Trasforma tutti i simboli di una stringa selezionata in minuscolo per posizione
StringToUpper	Trasforma tutti i simboli di una stringa selezionata in maiuscole, per posizione
StringTrimLeft	Taglia caratteri di alimentazione di linea, spazi e le schede nella parte sinistra della stringa
StringTrimRight	Taglia i caratteri di alimentazione di linea, spazi e le schede nella parte destra della stringa

StringAdd

La funzione aggiunge una sottostringa alla fine di una stringa.

```
bool  StringAdd(
   string&  string_var,        // stringa, al quale aggiungiamo
   string   add_substring      // stringa, che viene aggiunta
   );
```

Parametri

string_var

[in] [out] Stringa, a cui si aggiunge un'altra.

add_substring

[in] Stringa che viene aggiunta alla fine di una stringa di origine.

Valore restituito

In caso di successo restituisce true, altrimenti false. Al fine di ottenere un codice di errore, deve essere chiamata la funzione GetLastError().

Esempio:

```
voidOnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- primo metodo
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("tempo per 'c = a + b' = ",(stop-start)," millisecondi, i = ",i);

//--- second method
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- third method
   start=GetTickCount();
   a="a"; // re-inizializza la variabile a
   for(i=0;i<length;i++)
```

```
      {
       StringConcatenate(c,a,b);
      }
   stop=GetTickCount();
   Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i = ",i
  }
```

Vedi anche

StringConcatenate, StringSplit, StringSubstr

StringBufferLen

La funzione restituisce la dimensione del buffer allocato per la stringa.

```
int  StringBufferLen(
   string  string_var      // stringa
   )
```

Parametri

string_var

[in] String.

Valore restituito

Il valore 0 significa che la stringa è costante e la dimensione del buffer non può essere modificata. -1 significa che la stringa appartiene al terminale client, e la modifica del contenuto del buffer può avere risultati indeterminati.

Esempio:

```
voidOnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("prima: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("dopo: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

Vedi anche

StringAdd, StringInit, StringLen, StringFill

StringCompare

La funzione confronta due stringhe e restituisce il risultato del confronto sottoforma di numero intero.

```
int  StringCompare(
   const string&  string1,                 // la prima stringa nel confronto
   const string&  string2,                 // la seconda stringa nel confronto
   bool           case_sensitive=true      // selezione modalità di sensibilità maiusc
   );
```

Parametri

string1

[in] la prima stringa.

string2

[in] la seconda stringa.

case_sensitive=true

[in] Modalità di selezione sensibilità maiuscole/minuscole. Se è vero, allora "A">"a". Se è falso, allora "A"="a". Per impostazione predefinita, il valore è uguale a true.

Valore restituito

- -1 (meno uno), se string1<string2
- 0 (zero), se string1=string2
- 1 (uno), se string1>string2

Nota

Le stringhe sono confrontate simbolo per simbolo, i simboli sono confrontati in ordine alfabetico secondo la tabella codici corrente.

Esempio:

```
voidOnStart()
  {
//--- chi è più grande - apple oppure home?
   string s1="Apple";
   string s2="home";

//--- confronta il case sensitive
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("Confronto case sensitive: %s > %s",s1,s2);
   else
     {
      if(result1<0)PrintFormat("Confronto case sensitive: %s < %s",s1,s2);
      else PrintFormat("Confronto case sensitive: %s = %s",s1,s2);
     }

//--- confronto case-insensitive
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("Confronto case insensitive: %s > %s",s1,s2);
   else
```

```
      {
       if(result2<0)PrintFormat("Confronto case insensitive: %s < %s",s1,s2);
       else PrintFormat("Confronto case insensitive: %s = %s",s1,s2);
      }
/* Risultato:
    Confronto case-sensitive: Apple < home
    Confronto case-insensitive: Apple < home
*/
  }
```

Vedi anche

Tipo Stringa, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Uso del Codepage

StringConcatenate

La funzione, forma una serie di parametri passati e restituisce la grandezza della stringa formata. I parametri possono essere di qualsiasi tipo. Il numero di parametri non può essere minore di 2 o più di 64.

```
int  StringConcatenate(
   string&  string_var,    // stringa da formare
   void argument1          // primo parametro di un qualsiasi tipo semplice
   void argument2          // secondo parametro di un qualsiasi tipo semplice
   ...                     // parametro successivo di qualsiasi tipo semplice
   );
```

Parametri

string_var

[out] Stringa che verrà formata come risultato della concatenazione.

argumentN

[in] Qualsiasi valore separato da virgola. Da 2 a 63 parametri di qualsiasi tipo semplice.

Valore restituito

Restituisce la lunghezza della stringa, formata dalla concatenazione di parametri trasformati in tipo stringa. I parametri vengono trasformati in stringhe secondo le stesse regole Print() e Comment().

Vedi anche

StringAdd, StringSplit, StringSubstr

StringFill

Riempie la stringa selezionata con simboli specificati.

```
bool  StringFill(
   string&   string_var,       // stringa da riempire
   ushort    character         // simbolo che riempirà la stringa
   );
```

Parametri

string_var

[in][out] Stringa, che verrà compilata dal simbolo selezionato.

character

[in] Simbolo, con la quale la stringa verrà compilata.

Valore restituito

In caso di successo restituisce true, in caso contrario - false. Per ottenere il codice di errore chiamare GetLastError().

Nota

La compilazione di una stringa in luogo significa che i simboli vengono inseriti direttamente nella stringa senza le operazioni transitorie di creazione di nuova stringa o la copia. Ciò consente di risparmiare il tempo di funzionamento.

Esempio:

```
voidOnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// Risultato
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

Vedi anche

StringBufferLen, StringLen, StringInit

StringFind

Ricerca di una sottostringa in una stringa.

```
int  StringFind(
   string  string_value,        // stringa in cui si fa ricerca
   string  match_substring,     // ciò che viene cercato
   int     start_pos=0          // da quale posizione parte la ricerca
   );
```

Parametri

string_value

[in] Stringa, in cui la ricerca viene effettuata.

match_substring

[in] sottostringa cercata.

start_pos=0

[in] Posizione nella stringa, da dove viene avviata la ricerca.

Valore restituito

Restituisce il numero di posizione in una stringa, da dove inizia la sottostringa ricercata, ooppure restituisce -1, se la sottostringa non viene trovata.

Vedi anche

StringSubstr, StringGetCharacter, StringLen, StringLen

StringGetCharacter

Restituisce il valore di un simbolo, che si trova nella posizione specificata di una stringa.

```
ushort  StringGetCharacter(
   string  string_value,     // stringa
   int     pos               // posizione del simbolo nella stringa
   );
```

Parametri

string_value

[in] String.

pos

[in] Posizione del simbolo nella stringa. Può essere da 0 a StringLen(testo) -1.

Valore restituito

Codice del symbolo o 0 in caso di errore. Per ottenere il codice di errore chiamare GetLastError().

Vedi anche

StringSetCharacter, StringBufferLen, StringLen, StringFill, StringInit, StringToCharArray, StringToShortArray

StringInit

Inizializza una stringa di simboli specificati e fornisce la grandezza della stringa specificata.

```
bool  StringInit(
   string&   string_var,       // stringa da inizializzare
   int       new_len=0,        // lunghezza richiesta della stringa dopo l'inizializza
   ushort    character=0       // simbolo, con il quale la stringa verrà riempita
   );
```

Parametri

string_var

[in] [out] Stringa che deve essere inizializzata e deinizializzata.

new_len=0

[in] Lunghezza della stringa dopo l'inizializzazione. Se la lunghezza = 0, esso deinizializza la stringa, cioè il buffer della stringa viene cancellato e l'indirizzo del buffer viene azzerato.

character=0

[in] Simbolo con cui riempire la stringa.

Valore restituito

In caso di successo restituisce true, in caso contrario - false. Per ottenere il codice di errore chiamare GetLastError().

Nota

Se il carattere = 0 e la lunghezza new_len>0, il buffer della stringa di lunghezza indicata sarà distribuito e riempito di zeri. La lunghezza della stringa sarà pari a zero, poiché l'intero buffer è riempito da terminatori di stringa.

Esempio:

```
voidOnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* Risultato
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

Vedi anche

StringBufferLen, StringLen

StringLen

Restituisce il numero di simboli in una stringa.

```
int  StringLen(
   string  string_value      // stringa
   );
```

Parametri

string_value

[in] Stringa per la quale calcolare la lunghezza.

Valore restituito

Numero di simboli in una stringa senza lo zero finale.

Vedi anche

StringBufferLen, StringTrimLeft, StringTrimRight, StringToCharArray, StringToShortArray

StringSetLength

Imposta una lunghezza specificata (in caratteri) per una stringa.

```
bool  StringSetLength(
   string&     string_var,       // stringa
   uint        new_length        // nuova lunghezza della stringa
   );
```

Parametri

string_var

[in] [out] Stringa, per la quale è necessario impostare una nuova lunghezza in caratteri.

new_capacity

[in] Lunghezza della stringa richiesta in caratteri. Se new_length è inferiore alla grandezza corrente, i caratteri eccessivi vengono scartati.

Valore di ritorno

In caso di esecuzione riuscita, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError().

Nota

La funzione StringSetLength() non modifica la grandezza del buffer allocato per una stringa.

Guarda anche

StringLen, StringBufferLen, StringReserve StringInit, StringSetCharacter

StringReplace

Esso sostituisce tutte le sottostringhe trovate di una stringa da una sequenza di insieme di simboli.

```
int  StringReplace(
   string&         str,              // la stringa in cui le sottostringhe verranno s
   const string    find,             // la sottostringa ricercata
   const string    replacement       // la sottostringa che verrà inserita nelle posi
   );
```

Parametri

str

[in] [out] La stringa in cui si sta andando a sostituire le sottostringhe.

find

[in] La sottostringa desiderata, da sostituire.

replacement

[in] La stringa che verrà inserita al posto di quella trovata.

Valore restituito

La funzione restituisce il numero di sostituzioni in caso di successo, altrimenti -1. Per ottenere un codice di errore chiamare la funzione GetLastError().

Nota

Se la funzione è stata eseguita correttamente, ma non sono state fatte sostituzioni (la sottostringa da sostituire non è stata trovata), essa restituisce 0.

L'errore può derivare da parametri *str* o *find* errati (stringa vuota o non inizializzata, vedere StringInit()). Inoltre, l'errore si verifica se la memoria non è sufficiente per completare la sostituzione.

Esempio:

```
  string text="The quick brown fox jumped over the lazy dog.";
  int replaced=StringReplace(text,"quick","slow");
  replaced+=StringReplace(text,"brown","black");
  replaced+=StringReplace(text,"fox","bear");
  Print("Rimpiazzato: ", replaced,". Result=",text);

//  Risultato
//  Rimpiazzato: 3. Risultato=The slow black bear jumped over the lazy dog.
//
```

Vedi anche

StringSetCharacter(), StringSubstr()

StringReserve

Riserva il buffer di una grandezza specificata per una stringa, in memoria.

```
bool  StringReserve(
   string&     string_var,       // stringa
   uint        new_capacity      // grandezza del buffer per la memorizzazione di una s
   );
```

Parametri

string_var

[in] [out] Stringa per cui dovrebbe cambiare la grandezza del buffer.

new_capacity

[in] Grandezza del buffer richiesta per una stringa. Se la grandezza new_capacity è inferiore alla lunghezza della stringa, la grandezza del buffer corrente non cambia.

Valore di ritorno

In caso di esecuzione riuscita, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError().

Nota

Generalmente, la grandezza della stringa non è uguale alla grandezza del buffer inteso per memorizzare la stringa. Quando si crea una stringa, il buffer appropriato viene solitamente assegnato con un margine. La funzione StringReserve() consente di gestire la grandezza del buffer e specificare la grandezza ottimale per le operazioni future.

Diversamente daStringInit(), la funzione StringReserve() non modifica il contenuto della stringa e non la riempie di caratteri.

Esempio:

```
void OnStart()
  {
   string s;
//--- controlla la velocità dell'operazione senza usare StringReserve
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- ora, facciamo lo stesso usando StringReserve
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- controlla l'ora
   Print("Test con StringReserve passato per "+(string)msc_reserve+" msc");
   Print("Test senza StringReserve passato per "+(string)msc_no_reserve+" msc");
```

```
/* Risultato:
    Test con StringReserve passato per 50 msc
    Test senza StringReserve passato per 121 msc
*/
  }
```

Guarda anche

StringBufferLen, StringSetLength, StringInit, StringSetCharacter

StringSetCharacter

Restituisce una copia di una stringa con un carattere modificato in una posizione specificata.

```
bool  StringSetCharacter(
   string&   string_var,       // stringa
   int       pos,              // posizione
   ushort    character         // carattere
   );
```

Parametri

string_var

[in][out] Stringa.

pos

[in] Posizione del carattere nella stringa. Può essere da 0 a StringLen(text).

character

[in] Codice Unicode del simbolo.

Valore restituito

In caso di successo restituisce true, altrimenti false. Al fine di ottenere un codice di errore, deve essere chiamata la funzione GetLastError().

Nota

Se pos è minore della lunghezza della stringa ed il valore del codice del simbolo = 0, la stringa è tagliata (ma la grandezza del buffer, distribuita per la stringa rimane invariata). La lunghezza della stringa diventa uguale a pos.

Se *pos* è uguale alla lunghezza della stringa, il simbolo specificato viene aggiunto alla fine della stringa, e la lunghezza è allargata di uno.

Esempio:

```
voidOnStart()
  {
   string str="0123456789";
   Print("prima: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- aggiunge il valore zero nel mezzo
   StringSetCharacter(str,6,0);
   Print("after: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- aggiungere simboli alla fine
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("aggiunta: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* Risultato
```

```
   prima: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
    dopo: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
   aggiunta: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

Vedi anche

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

StringSplit

Ottiene sottostringhe da un separatore specificato dalla stringa specificata, restituisce il numero di sottostringhe ottenute.

```
int  StringSplit(
   const string   string_value,       // La stringa nel quale cercare
   const ushort   separator,          // Il separatore, che viene usato per cercare le
   string         & result[]          // Un array passato per riferimento per ottenere
   );
```

Parametri

string_value

[in] La stringa da cui è necessario ottenere sottostringhe. La stringa non cambierà.

pos

[in] Il codice del carattere separatore. Per ottenere il codice, è possibile utilizzare la funzione StringGetCharacter ().

result[]

[out] L' array di stringhe in cui si trovano le sottostringhe ottenute.

Valore restituito

Il numero di sottostringhe nell'array result[]. Se il separatore non viene trovato nella stringa passata, una sola stringa di origine verrà inserita nell'array.

Se string_value è vuoto o NULL, la funzione restituisce zero. In caso di errore la funzione restituisce -1. Per ottenere il codice d' errore, chiamare la funzione GetLastError().

Esempio:

```
string to_split="_life_is_good_"; // Una stringa per dividere in sottostringhe
   string sep="_";                // Un separatore come carattere
   ushort u_sep;                  // Il codice del carattere separatore
   string result[];               // Un array per ottenere le stringhe
   //--- Ottiene il codice separatore
   u_sep=StringGetCharacter(sep,0);
   //--- Divide la stringa in sottostringhe
   int k=StringSplit(to_split,u_sep,result);
   //--- Mostra un commento
   PrintFormat("Stringhe ottenute: %d. Usato il separatore '%s' con il codice %d",k,se
   //--- Ora da in output tutte le stringhe ottenute
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

Vedi anche

StringReplace(), StringSubstr(), StringConcatenate()

StringSubstr

Estrae una sottostringa da una stringa di testo a partire dalla posizione specificata.

```
string  StringSubstr(
   string  string_value,      // stringa
   int     start_pos,         // posizione da cui iniziare
   int     length=-1          // lunghezza della stringa estratta
   );
```

Parametri

string_value

[in] Stringa da quale estrarre la sottostringa.

start_pos

[in] Posizione iniziale della sottostringa. Può essere da 0 a StringLen(testo) -1.

length=-1

[in] Lunghezza della sottostringa estratta. Se il valore del parametro è uguale a -1 o il parametro non è impostato, la stringa verrà estratta dalla posizione indicata fino a fine stringa.

Valore restituito

Copia della sottostringa estratta, se possibile. In caso contrario, restituisce una stringa vuota.

Vedi anche

StringSplit, StringFind, StringGetCharacter

StringToLower

Trasforma tutti i simboli di una stringa selezionata in minuscolo per posizione.

```
bool  StringToLower(
   string&  string_var      // stringa da elaborare
   );
```

Parametri

string_var

[in][out] Stringa.

Valore restituito

In caso di successo restituisce true, in caso contrario - false. Per ottenere il codice di errore chiamare GetLastError().

Vedi anche

StringToUpper, StringTrimLeft, StringTrimRight

StringToUpper

Trasforma tutti i simboli di una stringa selezionata in maiuscole per posizione.

```
bool  StringToUpper(
   string&  string_var      // stringa da elaborare
   );
```

Parametri

string_var

[in][out] Stringa.

Valore restituito

In caso di successo restituisce true, in caso contrario - false. Per ottenere il codice di errore chiamare GetLastError().

Vedi anche

StringToLower, StringTrimLeft, StringTrimRight

StringTrimLeft

La funzione taglia caratteri di avanzamento di linea, gli spazi e le schede nella parte sinistra della stringa fino al primo simbolo significativo. La stringa viene modificata nel posto.

```
int  StringTrimLeft(
   string&  string_var      // stringa da tagliare
   );
```

Parametri

string_var

[in][out] La stringa che viene tagliata dalla sinistra.

Valore restituito

Restituisce il numero di simboli tagliati.

Vedi anche

StringTrimRight, StringToLower, StringToUpper

StringTrimRight

La funzione taglia caratteri di avanzamento di linea, gli spazi e le schede nella parte destra della stringa fino al primo simbolo significativo. La stringa viene modificata nel posto.

```
int  StringTrimRight(
   string&  string_var      // stringa da tagliare
   );
```

Parametri

string_var

[in][out] La stringa che viene tagliata dalla destra.

Valore restituito

Restituisce il numero di simboli tagliati.

Vedi anche

StringTrimLeft, StringToLower, StringToUpper

Data ed Ora

Questo è il gruppo di funzioni per lavorare con dati di tipo datetime (un numero intero che rappresenta il numero di secondi trascorsi dall'ora 0 del 1 gennaio 1970).

Per organizzare contatori e timers ad alta risoluzione, utilizzare la funzione GetTickCount(), che produce valori in millisecondi.

Funzione	Azione
TimeCurrent	Restituisce l'ultimo orario conosciuto di server (orario della ricevuta dell'ultima quotazione) nel formato datetime
TimeTradeServer	Restituisce l'ora attuale del calcolo del server di trade
TimeLocal	Restituisce l'ora locale del computer in formato datetime
TimeGMT	Restituisce GMT in formato datetime con l'ora legale dell' orario locale del computer, in cui il terminale client è in esecuzione
TimeDaylightSavings	Restituisce il segno di switch dell'ora legale
TimeGMTOffset	Restituisce la corrente differenza tra l'orario GMT e l'ora del computer locale in secondi, tenendo in considerazione l'interruttore DST
TimeToStruct	Converte un valore datetime in una variabile di tipo struttura MqlDateTime
StructToTime	Converte una variabile di tipo struttura MqlDateTime in un valore datetime

TimeCurrent

Returns the last known server time, time of the last quote receipt for one of the symbols selected in the "Market Watch" window. Nell'handler OnTick(), questa funzione restituisce l'orario dell'ultimo tick handler ricevuto. In altri casi (ad esempio, la chiamata negli handlers OnInit(), OnDeinit(), OnTimer() e così via), questa è l' orario della ricezione dell' ultima quotazione per qualsiasi simbolo disponibile nella finestra "Market Watch", l'orario indicato nel titolo di questa finestra. Il valore dell'orario è formato su un trade server e non dipende dalle impostazioni di orario sul tuo computer. Ci sono due varianti della funzione.

Chiamata senza parametri

```
datetime  TimeCurrent();
```

Chiama con parametro di tipo MqlDateTime

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // variabile di tipo struttura
   );
```

Parametri

dt_struct

[out] MqlDateTime variabile di tipo struttura.

Valore restituito

Value of datetime type

Nota

Se la variabile MqlDateTime di tipo struttura è stata passata come parametro, viene riempita di conseguenza.

Per organizzare contatori e timers ad alta risoluzione, utilizzare la funzione GetTickCount(), che produce valori in millisecondi.

During testing in the strategy tester, TimeCurrent() is simulated according to historical data.

TimeTradeServer

Restituisce l'orario corrente calcolato del server di trade. A differenza di TimeCurrent(), il calcolo del valore di tempo viene eseguito nel terminale cliente e dipende dalle impostazioni dell'ora del computer. Ci sono due varianti della funzione.

Chiamata senza parametri

```
datetime  TimeTradeServer();
```

Chiama con parametro di tipo MqlDateTime

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // Variabile di tipo struttura
   );
```

Parametri

dt_struct

[out] Variabile di tipo struttura MqlDateTime.

Valore restituito

Value of datetime type

Nota

Se la variabile MqlDateTime di tipo struttura è stata passata come parametro, viene riempita di conseguenza.

Per organizzare contatori e timers ad alta risoluzione, utilizzare la funzione GetTickCount(), che produce valori in millisecondi.

During testing in the strategy tester, TimeTradeServer() is always equal to TimeCurrent() simulated server time.

TimeLocal

Restituisce l'ora locale di un computer, in cui il terminale client è in esecuzione. Ci sono due varianti della funzione.

Chiamata senza parametri

```
datetime  TimeLocal();
```

Chiama con parametro di tipo MqlDateTime

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // Variabile di tipo struttura
   );
```

Parametri

dt_struct

[out] Variabile di tipo struttura MqlDateTime.

Valore restituito

Value of datetime type

Nota

Se la variabile MqlDateTime di tipo struttura è stata passata come parametro, viene riempita di conseguenza.

Per organizzare contatori e timers ad alta risoluzione, utilizzare la funzione GetTickCount(), che produce valori in millisecondi.

During testing in the strategy tester, TimeLocal() is always equal to TimeCurrent() simulated server time.

TimeGMT

Restituisce il GMT, che viene calcolato tenendo conto del DST switch dall'ora locale del computer in cui il terminale client è in esecuzione. Ci sono due varianti della funzione.

Chiamata senza parametri

```
datetime  TimeGMT();
```

Chiama con parametro di tipo MqlDateTime

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // Variabile di tipo struttura
   );
```

Parametri

dt_struct

[out] Variabile di tipo struttura MqlDateTime.

Valore restituito

Value of datetime type

Nota

Se la variabile MqlDateTime di tipo struttura è stata passata come parametro, viene riempita di conseguenza.

Per organizzare contatori e timers ad alta risoluzione, utilizzare la funzione GetTickCount(), che produce valori in millisecondi.

During testing in the strategy tester, TimeGMT() is always equal to TimeTradeServer() simulated server time.

TimeDaylightSavings

Restituisce la correzione per l'ora legale in secondi, quando viene fatto il passaggio al periodo estivo. Dipende dalle impostazioni dell'ora del computer.

```
int  TimeDaylightSavings();
```

Valore restituito

Se lo switch per l'inverno (standard) Il tempo è stato fatto, restituisce 0.

TimeGMTOffset

Restituisce la corrente differenza tra l'orario GMT e l'ora del computer locale in secondi, tenendo conto dello switch per l'inverno o estate. Dipende dalle impostazioni dell'ora del computer.

```
int  TimeGMTOffset();
```

Valore restituito

Il valore di tipo int, che rappresenta la differenza di corrente tra l' ora GMT e l'ora locale del computer TimeLocal() in secondi.

```
TimeGMTOffset() =  TimeGMT() - TimeLocal()
```

TimeToStruct

Converte un valore di tipo datetime (numero di secondi dal 01.01.1970) in una variabile struttura MqlDateTime.

```
bool  TimeToStruct(
   datetime      dt,            // date ed ora
   MqlDateTime&  dt_struct      // struttura di adozione dei valori
   );
```

Parametri

dt

[in] Valori data da convertire.

dt_struct

[out] Variabile di struttura tipo MqlDateTime.

Valore restituito

True se avviene con successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

StructToTime

Converte una variabile di struttura MqlDateTime in un valore di tipo datetime e restituisce il valore risultante.

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // struttura della data e dell'ora
   );
```

Parametri

dt_struct

[in] Variabile della struttura di tipo MqlDateTime.

Valore restituito

Il valore di tipo datetime contenente il numero di secondi dal 01.01.1970.

Informazioni account

Funzioni che restituiscono i parametri del corrente account.

Funzione	Azione
AccountInfoDouble	Restituisce un valore di tipo double della corrispondente proprietà dell' account
AccountInfoInteger	Restituisce un valore di tipo integer (bool, int o long) della corrispondente proprietà dell' account
AccountInfoString	Restituisce un valore di tipo stringa della corrispondente proprietà dell'account

AccountInfoDouble

Restituisce il valore della corrispondente proprietà dell' account.

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Il valore può essere uno dei valori di ENUM_ACCOUNT_INFO_DOUBLE.

Valore restituito

Valore di tipo double.

Esempio:

```
voidOnStart()
  {
//--- mostra tutte le informazioni disponibili dalla funzione AccountInfoDouble()
   printf("ACCOUNT_BALANCE =  %G",AccountInfoDouble(ACCOUNT_BALANCE));
   printf("ACCOUNT_CREDIT =  %G",AccountInfoDouble(ACCOUNT_CREDIT));
   printf("ACCOUNT_PROFIT =  %G",AccountInfoDouble(ACCOUNT_PROFIT));
   printf("ACCOUNT_EQUITY =  %G",AccountInfoDouble(ACCOUNT_EQUITY));
   printf("ACCOUNT_MARGIN =  %G",AccountInfoDouble(ACCOUNT_MARGIN));
   printf("ACCOUNT_MARGIN_FREE =  %G",AccountInfoDouble(ACCOUNT_MARGIN_FREE));
   printf("ACCOUNT_MARGIN_LEVEL =  %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));
   printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));
   printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));
  }
```

Vedi anche

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

AccountInfoInteger

Restituisce il valore delle proprietà dell'account.

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Il valore può essere uno dei valori di ENUM_ACCOUNT_INFO_INTEGER.

Valore restituito

Valore di tipo long.

Nota

La proprietà deve essere uno dei tipi bool, int o long.

Esempio:

```
voidOnStart()
  {
//--- mostra tutte le informazioni disponibili ddella funzione AccountInfoInteger()
   printf("ACCOUNT_LOGIN =  %d",AccountInfoInteger(ACCOUNT_LOGIN));
   printf("ACCOUNT_LEVERAGE =  %d",AccountInfoInteger(ACCOUNT_LEVERAGE));
   bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
   bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
   ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(ACCOU
   ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoIntege

//--- Informa sulla possibilità di eseguire una operazione di trade
   if(thisAccountTradeAllowed)
      Print("Trading permesso per questo account");
   else
      Print("Trading proibito per questo account!");

//--- Rivela se è possibile fare trading su questo account, da Expert Advisors
   if(EATradeAllowed)
      Print("Trading da Expert Advisors, permesso su questo account");
   else
      Print("Trading da Expert Advisors, proibito su questo account!");

//--- Rivela il tipo di account
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("Questo è un account demo");
         break;
```

```
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("Questo è un account di competizione");
         break;
      default:Print("Questo è un account reale!");
     }

//--- Rivela il livello di modalità StopOut
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("Il livello di StopOut è specificato in percentuale");
         break;
      default:Print("Il livello di StopOut è specificato in termini monetari");
     }
  }
```

Vedi anche

Informazioni account

AccountInfoString

Restituisce il valore della corrispondente proprietà dell' account.

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Il valore può essere uno dei valori ENUM_ACCOUNT_INFO_STRING.

Valore restituito

Valore di tipo string.

Esempio:

```
voidOnStart()
  {
//--- Mostra tutte le informazioni disponibili dalla funzione AccountInfoString()
   Print("Il nome del broker = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Valuta di deposito = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Nome Cliente = ",AccountInfoString(ACCOUNT_NAME));
   Print("Il nome del trade server = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

Vedi anche

Informazioni account

Verifica Stato

Funzione che restituisce i parametri del corrente stato del terminale client

Funzione	Azione
GetLastError	Restituisce l'ultimo errore
IsStopped	Restituisce true, se un programma mql5 è stato comandato di fermarsi dal suo funzionamento.
UninitializeReason	Restituisce il codice della ragione per la deinizializzazione
TerminalInfoInteger	Restituisce un valore integer della corrispondente proprietà dell'ambiente del programma mql5.
TerminalInfoDouble	Restituisce un valore double di una proprietà corrispondente dell'ambiente programma MQL5
TerminalInfoString	Restituisce il valore stringa della corrispondente proprietà dell'ambiente del programma mql5.
MQLInfoInteger	Restituisce il valore integer della corrispondente proprietà del programma mql5 che sta girando
MQLInfoString	Restituisce il valore stringa della corrispondente proprietà del programma mql5 che sta girando
Symbol	Restituisce il nome del simbolo del corrente grafico
Period	Restituisce il timeframe del corrente grafico
Digits	Restituisce il numero di cifre decimali determinando l'accuratezza del valore prezzo del corrente simbolo del grafico
Point	Restituisce la grandezza in punti del simbolo corrente nella valuta di quotazione

GetLastError

Restituisce il contenuto della variabile del sistema _LastError.

```
int  GetLastError();
```

Valore restituito

Restituisce il numero dell'ultimo errore che si è verificato durante l'esecuzione del programma mql5.

Nota

Dopo la chiamata della funzione, il contenuto di _LastError non viene resettato. Per resettare questa variabile, bisogna chiamare ResetLastError().

Vedi anche

Codici di Ritorno del Trade Server

IsStopped

Verifica lo spegnimento forzato del programma mql5.

```
bool  IsStopped();
```

Valore restituito

Restituisce true, se la variabile di sistema _StopFlag contiene un valore che non sia 0. Un valore non-zero viene scritto in _StopFlag, se un programma mql5 è stato comandato a completare il suo funzionamento. In questo caso, bisogna immediatamente terminare il programma, altrimenti il programma verrà completamente forzato dall'esterno dopo 3 secondi.

UninitializeReason

Restituisce il codice del motivo di deinizializzazione.

```
int  UninitializeReason();
```

Valore restituito

Restituisce il valore di _UninitReason che viene formato prima che OnDeinit() venga chiamato. Il valore dipende dalle ragioni che hanno portato alla deinizializzazione.

TerminalInfoInteger

Restituisce il valore della corrispondente proprietà dell'ambiente di programma mql5.

```
int  TerminalInfoInteger(
   int  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Può essere uno dei valori dell'enumerazione ENUM_TERMINAL_INFO_INTEGER.

Valore restituito

Valore di tipo int.

TerminalInfoDouble

Restituisce il valore di una proprietà corrispondente dell'ambiente programma MQL4.

```
double  TerminalInfoDouble(
   int  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Può essere uno dei valori dell'enumerazione ENUM_TERMINAL_INFO_DOUBLE.

Valore restituito

Valore di tipo double.

TerminalInfoString

la funzione resituisce il valore della proprietà corrispondente dell'ambiente di programma mql5. La proprietà dev'essere di tipo stringaThe property must be of string type.

```
string  TerminalInfoString(
   int  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Forse uno dei valori dell'enumerazione ENUM_TERMINAL_INFO_STRING.

Valore restituito

Valore di tipo stringa.

MQLInfoInteger

Restituisce il valore della corrispondente proprietà di un programma mql5 che sta girando.

```
int  MQLInfoInteger(
   int  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Può essere uno dei valori dell'enumerazione ENUM_MQL_INFO_INTEGER.

Valore restituito

Valore di tipo int.

MQLInfoString

Restituisce il valore della corrispondente proprietà di un programma mql5 che sta girando.

```
string  MQLInfoString(
   int  property_id      // Identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Può essere uno dei valori dell'enumerazione ENUM_MQL_INFO_STRING.

Valore restituito

Valore di tipo stringa.

Symbol

Restituisce il nome del simbolo del corrente grafico.

```
string  Symbol();
```

Valore restituito

Valore della variabile di sistema _Symbol, che memorizza il nome del corrente simbolo del grafico.

Nota

A differenza di Expert Advisors, indicatori e script, i servizi non sono associati ad uno specifico chart. Perciò, Simbol() restituisce una stringa vuota ("") per un servizio.

Period

Restituisce il timeframe del corrente grafico.

```
ENUM_TIMEFRAMES  Period();
```

Valore restituito

Il contenuto della variabile _Period che contiene il valore del timeframe del corrente grafico. Il valorep può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES.

Nota

A differenza di Expert Advisors, indicatori e script, i servizi non sono associati ad uno specifico chart. Perciò, Period() restituisce 0 per un servizio.

Vedi anche

PeriodSeconds, Timeframes del Grafico, Data ed Ora, Visibilità degli Oggetti

Digits

Restituisce il numero di cifre decimali determinando l'accuratezza del prezzo del corrente periodo del grafico.

```
int  Digits();
```

Valore restituito

Il valore della variabile _Digits che memorizza il numero di cifre decimali determinando l'accuratezza del prezzo del corrente simbolo del grafico.

Point

Restituisce la grandezza punti del corrente simbolo, nella valuta di quotazione.

```
double  Point();
```

Valore restituito

Il valore della variabile _Point che memorizza la grandezza in punti del corrente simbolo nella valuta di quotazione.

Gestione degli Eventi

Il linguaggio MQL5 fornisce la gestione di determinati eventi predefiniti. Le funzioni per la gestione di questi eventi (event handling) dovrebbero essere definite in un programma MQL5: nome funzione, tipo di ritorno, un insieme di parametri (se presenti), ed i loro tipi dovrebbero corrispondere strettamente alla descrizione di una funzione di gestione eventi.

Il gestore eventi del terminale client utilizza i tipi di ritorno e parametro per identificare le funzioni che elaborano un evento. Se una determinata funzione ha alcuni parametri o un tipo di ritorno non corrispondente alle descrizioni sottostanti, tale funzione non può essere utilizzata per gestire un evento.

Funzione	Descrizione
OnStart	La funzione è chiamata quando si verifica l'evento Start per eseguire azioni impostate nello script
OnInit	La funzione è chiamata negli indicatori ed EA quando si verifica l'evento Init per inizializzare un programma MQL5 avviato
OnDeinit	La funzione è chiamata negli indicatori ed EA quando si verifica l'evento Deinit per de-inizializzare un programma MQL5 avviato
OnTick	La funzione è chiamata nell' EA quando si verifica l'evento NewTick per gestire una nuova quotazione
OnCalculate	La funzione è chiamata negli indicatori quando si verifica l'evento Calculate per gestire i cambiamenti dei dati di prezzo
OnTimer	La funzione è chiamata negli indicatori ed EA durante l'evento periodico Timer generato dal terminale a intervalli di tempo fissi
OnTrade	La funzione è chiamata nell' EA durante l'evento generato Trade al termine di un'operazione di trading su un trade server
OnTradeTransaction	La funzione è chiamata nell' EA quando si verifica l'evento TradeTransaction per elaborare un risultato di esecuzione di una richiesta di trade
OnBookEvent	La funzione è chiamata nell'EA quando si verifica l'evento BookEvent per elaborare i cambiamenti nel market depth
OnChartEvent	La funzione è chiamata in indicatori ed EA quando si verifica l'evento ChartEvent per elaborare le modifiche del grafico-chart effettuate da un utente o un programma MQL5
OnTester	La funzione è chiamata nell'EA quando si verifica l'evento Tester per eseguire le azioni necessarie dopo aver testato un EA sui dati della cronistoria
OnTesterInit	La funzione è chiamata nell' EA quando si verifica l'evento TesterInit per eseguire le azioni necessarie prima dell'ottimizzazione nel tester di strategia

Funzione	Descrizione
OnTesterDeinit	La funzione è chiamata nell'EA quando si verifica l'evento TesterDeinit dopo l'ottimizzazione EA nel tester di strategia
OnTesterPass	La funzione è chiamata nell' EA quando si verifica l'evento TesterPass per gestire l'arrivo di un nuovo frame di dati, durante l'ottimizzazione EA nel tester di strategia

Il terminale client invia eventi in entrata ai corrispondenti grafici-chart aperti. Inoltre, gli eventi possono essere generati da grafici-charts (eventi del chart) o programmi mql5 (eventi personalizzati). La generazione di eventi di creazione/eliminazione di oggetti grafici può essere abilitata/disabilitata impostando le proprietà del chart CHART_EVENT_OBJECT_CREATE e CHART_EVENT_OBJECT_DELETE. Ogni applicazione e grafico-chart mql5 ha una propria coda di eventi in cui vengono posizionati tutti gli eventi appena arrivati.

Un programma ottiene eventi solo dal grafico-chart su cui è in esecuzione. Tutti gli eventi vengono gestiti uno dopo l'altro nell'ordine di ricezione. Se la coda contiene già l'evento NewTick o questo evento è in fase di elaborazione, allora il nuovo evento NewTick non viene aggiunto alla coda dell'applicazione mql5. Allo stesso modo, se ChartEvent è già in una coda di programma mql5 o viene gestito un evento di questo tipo, allora un nuovo evento di questo tipo non viene inserito in coda. La gestione degli eventi del timer viene elaborata allo stesso modo - se l'evento Timer è già in coda o viene gestito, nessun nuovo evento timer viene impostato in coda.

Le code degli eventi hanno una grandezza limitata ma sufficiente, quindi l'overflow della coda è improbabile per un programma sviluppato correttamente. Quando la coda si sovraffolla, i nuovi eventi vengono eliminati senza essere inseriti in coda.

Si raccomanda vivamente di non utilizzare loop infiniti per gestire gli eventi. Eventuali eccezioni sono gli script che gestiscono un singolo evento Start.

Le Librerie non gestiscono alcun evento.

OnStart

La funzione è chiamata quando si verifica l'evento Start. La funzione è intesa all'esecuzione una tantum delle azioni implementate nello script. Ci sono due tipi di funzione.

La versione che restituisce il risultato

```
int  OnStart(void);
```

Valore di ritorno

Il valore di tipo int visualizzato nella scheda Journal.

La voce "script nome_script rimosso (codice risultato N)" viene creata nel journal del terminale dopo che è stata completata l'esecuzione di uno script. Qui N è un valore restituito dalla funzione OnStart().

La voce "servizio nome_servizio interrotto (codice risultato N)" viene creata nel journal del terminale dopo che è stata completata l'esecuzione di un servizio. Qui N è un valore restituito dalla funzione OnStart().

La chiamata OnStart() che restituisce il risultato dell'esecuzione è consigliata per l'uso poiché non solo consente di eseguire uno script o un servizio, ma restituisce anche un codice di errore o altri dati utili per analizzare il risultato dell'esecuzione del programma.

La versione senza un risultato restituito è lasciata solo per compatibilità con i vecchi codici. Non è raccomandata per l'uso

```
void  OnStart(void);
```

Nota

OnStart() è l'unica funzione per gestire gli eventi in script e servizi. Nessun altro evento viene inviato a questi programmi. A sua volta, l'evento Start non viene passato agli EA e agli indicatori personalizzati.

Script di esempio:

```
//--- macro per lavorare con i colori
#define XRGB(r,g,b)     (0xFF000000|(uchar(r)<<16)|(uchar(g)<<8)|uchar(b))
#define GETRGB(clr)     ((clr)&0xFFFFFF)
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imposta un colore di candela verso il basso
   Comment("Imposta un colore di candela verso il basso");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // aggiorna il chart immediatamente senza aspettare un nuovo tick
   Sleep(1000);   // pausa per 1 secondo per vedere tutte le modifiche
//--- imposta un colore di candela verso l'alto
   Comment("Imposta un colore di candela verso l'alto");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
```

```
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore di sfondo
   Comment("Imposta il colore di sfondo");
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore della linea Ask
   Comment("Imposta il colore della linea Ask");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore della linea Bid
   Comment("Imposta il colore della linea Bid");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
// --- imposta il colore di una barra verso il basso e una cornice di candela verso il
   Comment("Imposta il colore di una barra verso il basso e una cornice di candela ver
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore di una linea del chart e delle candele Doji
   Comment("Imposta il colore di una linea del grafico e dei candelabri Doji");
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore di una barra verso l'alto e di una cornice di candele verso l
   Comment("Imposta il colore di una barra verso l'alto e una cornice per candele vers
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore degli assi, della scala e della linea OHLC
   Comment("Imposta il colore degli assi, la scala e la linea OHLC");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta un colore griglia
   Comment("Imposta un colore griglia");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore dell'ultimo prezzo
   Comment("Imposta il colore dell'ultimo prezzo");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- imposta il colore dei livelli di ordine Stop Loss e Take Profit
   Comment("Imposta il colore dei livelli di ordine Stop Loss e Take Profit");
```

```
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
/ --- imposta il colore dei volumi e i livelli di entrata nel mercato
   Comment("Imposta il colore dei volumi e i livelli di entrata nel mercato");
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Restituisce un colore generato a caso                            |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

Guarda anche

Funzioni di gestione degli eventi(event handling), Esecuzione Programma, Eventi del terminale client

OnInit

La funzione è chiamata in indicatori ed EA quando si verifica l'evento Init. Viene utilizzato per inizializzare un programma MQL5 in esecuzione. Ci sono due tipi di funzione.

La **versione che restituisce il risultato**

```
int  OnInit(void);
```

Valore di ritorno

int (tipo di valore), zero significa inizializzazione riuscita.

La chiamata OnInit() che restituisce il risultato dell'esecuzione è consigliata per l'uso poiché non solo consente l'inizializzazione del programma, ma restituisce anche un codice di errore in caso di una chiusura anticipata del programma.

La versione senza un risultato restituito è lasciata solo per compatibilità con i vecchi codici. Non è raccomandata per l'uso

```
void  OnInit(void);
```

Nota

L'evento Init viene generato immediatamente dopo aver caricato un EA o un indicatore. L'evento non è generato per gli script. La funzione OnInit() viene utilizzata per inizializzare un programma MQL5. Se OnInit() ha un valore di ritorno di tipo int, il codice di ritorno diverso da zero indica l'inizializzazione fallita e genera l'evento Deinit con il codice del motivo di deinizializzazione REASON_INITFAILED.

La funzione OnInit() di tipo void significa sempre inizializzazione riuscita e non è raccomandato per l'uso.

Per ottimizzazione degli input dell' EA, si consiglia di utilizzare valori dall'enumerazione ENUM_INIT_RETCODE come codice di ritorno. Questi valori sono intesi a stabilire la gestione del processo di ottimizzazione, compresa la selezione dei più adatti agenti di testing. È possibile richiedere i dati sulla configurazione e sulle risorse degli agenti (numero di core, quantità di memoria libera, ecc.) utilizzando la funzione TerminalInfoInteger() durante l'inizializzazione di EA prima dell'avvio del test. Sulla base dei dati ottenuti, è possibile consentire l'uso dell'agente di testing o vietarlo dall'ottimizzazione di EA.

ID	Descrizione
INIT_SUCCEEDED	Inizializzazione riuscita, il test EA può essere continuato. Questo codice indica lo stesso valore zero: l'inizializzazione EA nel tester ha esito positivo.
INIT_FAILED	Inizializzazione fallita. Non ha senso continuare il test, a causa di errori inevitabili. Ad esempio, è impossibile creare un indicatore necessario per l'operazione EA. Il ritorno di questo valore equivale a restituire il valore diverso da zero - l'inizializzazione di EA nel tester non è riuscita.
INIT_PARAMETERS_INCORRECT	Designato per denotare un insieme errato di parametri di input da parte di un programmatore. Nella tabella di ottimizzazione

ID	Descrizione
	generale, la stringa dei risultati con questo codice di ritorno è evidenziata in rosso. Non viene eseguito un test per un tale insieme di ingressi EA. L'agente è pronto per ricevere una nuova attività. Quando questo valore viene ricevuto, il tester della strategia non trasferisce questa attività ad altri agenti per l'esecuzione ripetuta.
INIT_AGENT_NOT_SUITABLE	Nessun errore di esecuzione del programma durante l'inizializzazione. Tuttavia, per alcuni motivi, l'agente non è adatto a condurre un test. Ad esempio, non c'è abbastanza RAM, nessun Supporto OpenCL, eccetera. Dopo aver restituito questo codice, l'agente non riceve più attività fino alla fine di questa ottimizzazione.

Utilizzando OnInit() restituendo INIT_FAILED/INIT_PARAMETERS_INCORRECT nel tester ha alcune peculiarità che dovrebbero essere considerate quando si ottimizzano gli EA:

- l'insieme di parametri OnInit() restituito INIT_PARAMETERS_INCORRECT per è considerato inadatto per il test e non viene utilizzato per ottenere la successiva popolazione durante l'ottimizzazione genetica. Troppi set di parametri "scartati" possono portare a risultati errati durante la ricerca di parametri EA ottimali. L'algoritmo di ricerca presuppone che la funzione criterio di ottimizzazione sia regolare e non abbia spazi vuoti sull'intera moltitudine di parametri di input.
- se OnInit() restituisce INIT_FAILED, ciò significa che non è possibile avviare un test e che l'EA viene de-caricato dalla memoria dell'agente. L'EA viene nuovamente caricato per eseguire il passaggio successivo con una nuova serie di parametri. L'avvio del prossimo passaggio di ottimizzazione richiede molto più tempo rispetto a chiamare TesterStop().

Esempio di funzione OnInit() per un EA

```
//--- parametri di input
input int      ma_period=20; // periodo media mobile

//--- handle dell'indicatore utilizzato nell'EA
int indicator_handle;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- controlla la validità di ma_period
   if(ma_period<=0)
     {
      PrintFormat("Valore input ma_period non valido: %d",ma_period);
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- durante l'ottimizzazione
   if(MQLInfoInteger(MQL_OPTIMIZATION))
```

```
      {
       //--- controlla la RAM disponibile per l'agente
       int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
       if(available_memory_mb<2000)
         {
          PrintFormat("Memoria insufficiente per l'agente test: %d MB",
                      available_memory_mb);
          return (INIT_AGENT_NOT_SUITABLE);
         }
      }
//--- controlla l'indicatore
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
     {
      PrintFormat("Fallimento nel generare l'handle My_Indicator . Codice errore %d",
                  GetLastError());
      return (INIT_FAILED);
     }
//--- Inizializzazione EA completata
   return(INIT_SUCCEEDED);
  }
```

Guarda anche

OnDeinit, Funzioni di gestione degli eventi(event handling), Esecuzione Programma, Eventi del terminale client, Inizializzazione di variabili, Creazione ed eliminazione di oggetti

OnDeinit

La funzione è chiamata in indicatori ed EA quando si verifica l'evento Deinit. È usato per deinitializzare un programma MQL5 in esecuzione.

```
void  OnDeinit(
   const int  reason         // codice della ragione di deinizializzazione
   );
```

Parametri

reason

[in] Codice del motivo di deinitializzazione.

Valore di ritorno

Nessun valore di ritorno

Nota

L'evento Deinit è generato per EA e indicatori nei seguenti casi:

- prima di una nuova re-inizializzazione a causa del cambiamento di un simbolo o di un periodo del chart a cui è collegato il programma mql5;
- prima di una nuova inizializzazione a causa del cambiamento degli input;
- prima di decaricare un programma mql5.

Il parametro *reason* può avere i seguenti valori:

Costante	Valore	Descrizione
REASON_PROGRAM	0	L'EA ha smesso di funzionare chiamando la funzione ExpertRemove()
REASON_REMOVE	1	Programma rimosso da un chart
REASON_RECOMPILE	2	Programma ricompilato
REASON_CHARTCHANGE	3	Un simbolo o periodo chart sono cambiati
REASON_CHARTCLOSE	4	Chart chiuso
REASON_PARAMETERS	5	Input modificati da un utente
REASON_ACCOUNT	6	È stato attivato un altro account o si è verificata la riconnessione al trade server a causa di modifiche alle impostazioni dell'account
REASON_TEMPLATE	7	È stato applicato un altro modello di chart
REASON_INITFAILED	8	Il gestore OnInit() ha restituito un valore diverso da zero
REASON_CLOSE	9	Terminale chiuso

EA: i codici motivo di deinizializzazione possono essere ricevuti dalla funzione UninitializeReason() o dalla variabile predefinita _UninitReason.

Esempio di funzioni OnInit() e OnDeinit() per EA

```
input int fake_parameter=3;      // parametro inutile
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
// --- Ottieni il numero della build in cui è compilato il programma
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- il reset del codice motivazione (reason code) può anche essere ottenuto in OnIni
   Print(__FUNCTION__," Il codice motivo di deinitializzazione può essere ricevuto dur
//--- Il primo modo per ottenere un codice motivo di deinitializzazione
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Il secondo modo per ottenere un codice motivo di deinitializzazione
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di deinitializzazione dell' Expert                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Il primo modo per ottenere un codice motivo di deinitializzazione
   Print(__FUNCTION__," Codice di motivazione di Deinizializzazione = ",reason);
//--- Il secondo modo per ottenere un codice motivo di deinitializzazione
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Il terzo modo per ottenere un codice motivo di deinitializzazione
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
  }
//+------------------------------------------------------------------+
//| Restituisce una descrizione testuale del codice del motivo di deinitializzazione
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="L'account è cambiato";break;
      case REASON_CHARTCHANGE:
         text="Il simbolo o timeframe sono cambiati";break;
      case REASON_CHARTCLOSE:
         text="Il chart Chart è stato chiuso";break;
      case REASON_PARAMETERS:
         text="I parametri di Input sono stati cambiati";break;
      case REASON_RECOMPILE:
```

```
         text="il programma "+__FILE__+" è stato ricompilato";break;
      case REASON_REMOVE:
         text="il programma "+__FILE__+" è stato rimosso dal chart";break;
      case REASON_TEMPLATE:
         text="Il nuovo template è stato applicato al chart";break;
      default:text="Altra motivazione";
     }
//---
   return text;
  }
```

Guarda anche

OnInit, Funzioni di Event handling(gestione degli eventi), Esecuzione programma, Eventi del terminale client, Codici di motivazione di non inizializzazione, Scopo della visibilità e durata delle variabili, Creazione ed eliminazione di oggetti

OnTick

La funzione è chiamata nell' EA quando si verifica l'evento NewTick si verifica per gestire una nuova quotazione.

```
void  OnTick(void);
```

Valore di ritorno

Nessun valore di ritorno

Nota

L'evento NewTick viene generato solo per EA quando si riceve un nuovo tick per un simbolo del chart a cui è assegnato l' EA. Non ha senso definire la funzione OnTick() in un indicatore personalizzato o uno script poiché non viene generato un evento NewTick.

L'evento Tick è generato solo per EA, ma questo non significa che gli EA debbano presentare la funzione OnTick(), poiché gli eventi Timer, BookEvent e ChartEvent vengono generati anche per l'EA, oltre a NewTick.

Tutti gli eventi vengono gestiti uno dopo l'altro nell'ordine di ricezione. Se la coda contiene già l'evento NewTick o questo evento è in fase di elaborazione, allora il nuovo evento NewTick non viene aggiunto alla coda dell'applicazione mql5.

L'evento NewTick viene generato indipendentemente dal fatto che il trading automatico sia abilitato (pulsante AutoTrading). Il trading automatico disabilitato significa solo il divieto di inviare richieste trade da un EA. L'operazione EA non viene interrotta.

La disabilitazione del trading automatico premendo il pulsante AutoTrading non interrompe l'esecuzione corrente della funzione OnTick().

Esempio di EA con la sua intera logica di trading nella funzione OnTick()

```
//+------------------------------------------------------------------+
//|                                                   TradeByATR.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Esempio di trading EA in \"explosive\" direzione candela"
#property description "\"Explosive\" la candela ha la grandezza del corpo superiore a
#property description "The \"revers\"parametro inverte la direzione del segnale"

input double lots=0.1;        // volume in lotti
input double kATR=3;          // signal lunghezza candela in ATR
input int    ATRperiod=20;    // ATR indicator period
input int    holdbars=8;      // numero di barre per mantenere la posizione attiva
input int    slippage=10;     // slippage consentito
input bool   revers=false;    // inverto il segnale?
input ulong  EXPERT_MAGIC=0;  // EA MagicNumber
```

```
//--- per memorizzare l'handle dell'indicatore ATR
int atr_handle;
//--- qui memorizzeremo gli ultimi valori ATR e il corpo della candela
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- inizializza le variabili globali
   last_atr=0;
   last_body=0;
//--- imposta il volume corretto
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- crea l'handle dell'indicatore ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: impossibile creare iATR, codice di errore %d",__FUNCTION__,GetI
      return(INIT_FAILED);
     }
//--- inizializzazione EA di successo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di deinitializzazione dell' Expert                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- informa del codice di fine operazione EA
   Print(__FILE__,": Codice della ragione di deinizializzazione = ",reason);
  }
//+------------------------------------------------------------------+
//| Funzione tick Expert                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- segnale di trading
staticoint segnale = 0; // +1 significa un segnale di acquisto, -1 indica un segnale d
//--- controlla e chiude le vecchie posizioni aperte più di barre 'holdbars' fa
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- controlla una nuova barra
   if(isNewBar())
     {
      // --- controlla la presenza del segnale
      signal=CheckSignal();
```

```
      }
// --- se si apre una posizione netting, saltare il segnale - attendere fino a quando
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // esce dal gestore eventi NewTick e non entra nel mercato prima che app
     }
//--- per un conto hedging, ogni posizione viene tenuta e chiusa separatamente
   if(signal!=0)
     {
      //--- segnale buy
      if(signal>0)
        {
         PrintFormat("%s: Segnale Buy! Revers=%s",__FUNCTION__,string(revers));
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- segnale sell
      if(signal<0)
        {
         PrintFormat("%s: segnale Sell! Revers=%s",__FUNCTION__,string(revers));
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- Fine funzione OnTick
  }
//+------------------------------------------------------------------+
//| Controlla un nuovo segnale di trading                            |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 significa nessun segnale
   int res=0;
//--- ottiene il valore ATR su una penultima barra completa (l'indice della barra è 2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      // --- recupera i dati sull'ultima barra chiusa sull'array di tipo MqlRates
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         // --- calcola la misura del corpo della barra sull'ultima barra completa
         last_body=bar[0].close-bar[0].open;
// --- se il corpo dell'ultima barra (con indice 1) supera il precedente valore ATR (s
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // valore positivo per la candela verso l'alto
        }
```

```
      else
         PrintFormat("% s: impossibile ricevere l'ultima barra! Error",__FUNCTION__,Ge
     }
   else
      PrintFormat("% s: impossibile ricevere il valore dell'indicatore ATR! Error",__F
//--- se la modalità di trading inverso è abilitata
   res=revers?-res:res;  // inverte il segnale se necessario (restituisci -1 invece di
//--- restituisce un valore del segnale di trading
   return (res);
  }
//+------------------------------------------------------------------+
//| Restituisce 'true' quando appare una nuova barra                 |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // memorizza l'orario di apertura della barra corrente
//--- ottiene l'orario di apertura della barra zero
   datetime currbar_time=iTime(_Symbol,_Period,0);
// --- se l'orario di apertura cambia, è arrivata una nuova barra
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- visualizza i dati sull'orario di apertura di una nuova barra nel log
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- visualizza un messaggio con una nuova barra di apertura
         PrintFormat("%s: nuova barra su %s %s aperta a %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- recupera i dati sull'ultimo tick
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() fallito, errore = ",GetLastError());
         //--- mostra l'orario dell'ultimo tick fino ai millisecondi
         PrintFormat("L'ultimo tick era alle %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- abbiamo una nuova barra
      return (true);
     }
//--- nessuna nuova barra
   return (false);
  }
//+------------------------------------------------------------------+
//| Acquista ad un prezzo di mercato con un volume specificato       |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
```

```
//--- compra a prezzo di mercato
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Vendi ad un prezzo di mercato con un volume specificato          |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vendi a prezzo di mercato
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Chiusura le posizioni per hold time in barre                     |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // numero di posizioni aperte
//--- itera su posizioni aperte
   for(int i=total-1; i>=0; i--)
     {
      //--- parametri dell posizione
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- se la durata(lifetime) di una posizione è già grande, mentre MagicNumber e
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // abbassa il case del testo per la corretta formatt
         PrintFormat("Chiusura posizione #%d %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- imposta un tipo di ordine e invia una richiesta di trade
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Preparare e inviare una richiesta di trade                       |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulonc
```

```
  {
//--- dichiarare e inizializzare le strutture
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- richiesta parametri
   request.action   =TRADE_ACTION_DEAL;                     // tipi di operazioni di t
   request.position =pos_ticket;                            // ticket della posizione,
   request.symbol   =Symbol();                              // symbol
   request.volume   =volume;                                // volume
   request.type     =type;                                  // tipo di ordine
   request.price    =price;                                 // prezzo di trade
   request.deviation=slip;                                  // deviazione ammissibile
   request.magic    =magicnumber;                           // MagicNumber dell'ordine
//--- invia richiesta
   if(!OrderSend(request,result))
     {
      //--- mostra dati sul fallimento
      PrintFormat("OrderSend %s %s %.2f at %.5f errore %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- informa di un'operazione riuscita
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Guarda anche

Funzioni di gestione degli eventi, Esecuzione Programma, Eventi del terminale client, OnTimer, OnBookEvent, OnChartEvent

OnCalculate

La funzione è chiamata negli indicatori quando si verifica l'evento Calculate per l'elaborazione delle modifiche dei dati di prezzo. Ci sono due tipi di funzione. Solo uno di esse può essere utilizzata all'interno di un singolo indicatore.

Calcolo basato su matrice di dati

```
int  OnCalculate(
   const int        rates_total,       // grandezza array price[ ]
   const int        prev_calculated,   // numero di barre gestite alla precedente chia
   const int        begin,             // numero indice nell'array price[ ] da cui pa
   const double&    price[]            // array di valori per il calcolo
   );
```

Calcoli basati sulle timeseries del corrente timeframe

```
int  OnCalculate(
   const int        rates_total,       // grandezza delle timeseries input
   const int        prev_calculated,   // numero di barre gestite alla precedente chia
   const datetime&  time[],            // array Time
   const double&    open[],            // array Open
   const double&    high[],            // array High
   const double&    low[],             // array Low
   const double&    close[],           // array Close
   const long&      tick_volume[],     // array Tick Volume
   const long&      volume[],          // array Real Volume
   const int&       spread[]           // array Spread
   );
```

Parametri

rates_total

[in] Gradenzza della serie di prezzi[ ] o serie di input disponibili per l'indicatore per il calcolo. Nel secondo tipo di funzione, il valore del parametro corrisponde al numero di barre sul grafico-chart su cui è stato lanciato.

prev_calculated

[in] Contiene il valore restituito dalla funzione OnCalculate() durante la chiamata precedente. È progettato per saltare le barre che non sono state modificate dal lancio precedente di questa funzione.

begin

[in] Il valore dell'indice nell'array price[] da cui partono i dati significativi. Permette di saltare i dati mancanti o iniziali, per i quali non ci sono valori corretti.

price[]

[in] Array di valori per i calcoli. Una delle timeseries prezzo o un buffer indicatore calcolato può essere passato come array price[ ]. Il tipo di dati passati per il calcolo può essere definito usando la variabile predefinita _AppliedTo.

time{}

[in] Array con valori di tempo della barra aperta.

open[]

[in] Array con valori di prezzo Open.

high[]

[in] Array con valori di prezzo High.

low[]

[in] Array con valori di prezzo Low.

close[]

[in] Array con valori di prezzo Close.

tick_volume[]

[in] Array con valori di volume tick.

volume[]

[in] Array con valori di volume trade.

spread[]

[in] Array di valori spread per le barre.

Valore di ritorno

tipo int di valore da passare come parametro *prev_calculated* durante la prossima chiamata di funzione.

Nota

Se la funzione OnCalculate() è uguale a zero, nessun valore dell'indicatore viene visualizzato nel DataWindow del terminale client.

Se i dati del prezzo sono stati modificati dall'ultima chiamata della funzione OnCalculate() (è stato caricato uno storico più profondo o sono stati riempiti spazi vuoti nello storico), il valore del parametro di input *prev_calculated* è impostato a zero dal terminale stesso.

Per definire la direzione di indicizzazione negli array *time[]*, *open[]*, *high[]*, *low[]*, *close[]*, *tick_volume[]*, *volume[]* e *spread[]*, chiamare la funzione ArrayGetAsSeries(). Per non dipendere dai valori predefiniti, chiamare la funzione ArraySetAsSeries() per gli array con cui lavorare.

Quando si utilizza il primo tipo di funzione, un indicatore o timeseries necessari viene selezionato da un utente come array price[ ] nella scheda Parametri quando si avvia l'indicatore. Per fare ciò, specificare l'elemento necessario nell'elenco a discesa del campo "Applica a".

Per ottenere i valori dell'indicatore personalizzato da altri programmi mql5, viene usata la funzione iCustom(). Restituisce l'handle dell'indicatore per le operazioni successive. È anche possibile specificare l'array *price[ ]* richiesto o l'handle di un altro indicatore. Questo parametro deve essere passato per ultimo nell'elenco delle variabili di input di un indicatore personalizzato.

È necessario utilizzare la connessione tra il valore restituito dalla funzione OnCalculate() e il secondo parametro di input *prev_calculated*. Quando si chiama la funzione, il parametro *prev_calculated* contiene il valore restituito dalla funzione OnCalculate() durante la chiamata precedente. Ciò consente di implementare algoritmi di risparmio delle risorse per il calcolo di un indicatore

personalizzato al fine di evitare calcoli ripetitivi per le barre che non sono state modificate dal lancio precedente di questa funzione.

Indicatore di esempio

```
//+------------------------------------------------------------------+
//|                                          OnCalculate_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Calcolo dell'indicatore Momentum di esempio"

//---- Impostazioni dell'indicatore
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- input
input int MomentumPeriod=14; // periodo di Calcolo
//---- buffer indicatore
double    MomentumBuffer[];
//--- variabile globale per la memorizzazione del periodo di calcolo
int       IntPeriod;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione dell'indicatore personalizzato      |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- controlla il parametro di input
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("Il parametro Period ha un valore errato. Il seguente valore deve essere u
     }
   else
      IntPeriod=MomentumPeriod;
//---- buffers
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- nome dell'indicatore da visualizzare in DataWindow e sottofinestra
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
//--- imposta l'indice della barra da cui inizia il disegno
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- imposta 0.0 come valore vuoto che non è disegnato
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
```

```
//--- precisione dell'indicatore da visualizzare
   IndicatorSetInteger(INDICATOR_DIGITS,2);
  }
//+------------------------------------------------------------------+
//| Calcolo indicatore Momentum                                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,     // grandezza arrat price[ ]
                const int prev_calculated, // numero di barre precedentemente gestite
                const int begin,           // da dove partono i dati significativi
                const double &price[])     // array di valori per la gestione
  {
//--- posizione iniziale per i calcoli
   int StartCalcPosition=(IntPeriod-1)+begin;
// ---- se i dati di calcolo sono insufficienti
   if(rates_total<StartCalcPosition)
      return(0);  // esce con un valore zero - l'indicatore non viene calcolato
//--- inizia il disegno corretto
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- avvio calcoli, definisce la posizione di partenza
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- ciclo di calcolo principale
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- l'esecuzione OnCalcolate è completa. Restituisce il nuovo valore prev_calculated
   return(rates_total);
  }
```

Guarda anche

ArrayGetAsSeries, ArraySetAsSeries, iCustom, Funzioni di event handling, Avvio programmi, Eventi del terminale client, Accesso a timeseries ed indicatori

OnTimer

La funzione è chiamata nell' EA durante l'evento generato Timer dal terminale a intervalli di tempo fissi.

```
void  OnTimer(void);
```

Valore di ritorno

Nessun valore di ritorno

Nota

L'evento Timer viene periodicamente generato dal terminale client per un EA, che attiva il timer utilizzando la funzione EventSetTimer(). Di solito, questa funzione è chiamata nella funzione OnInit(). Quando l'EA smette di funzionare, il timer dovrebbe essere eliminato usando EventKillTimer(), che di solito è chiamato nella funzione OnDeinit().

Ogni Expert Advisor e ciascun indicatore funzionano con il proprio timer che riceve eventi esclusivamente da questo timer. Durante l'arresto dell'applicazione mql5, il timer viene eliminato forzatamente nel caso in cui sia stato creato ma non sia stato disattivato dalla funzione EventKillTimer().

Se è necessario ricevere eventi timer più frequentemente di una volta al secondo, utilizzare EventSetMillisecondTimer() per creare un timer ad alta risoluzione.

In generale, quando il periodo del timer viene ridotto, il tempo di test viene aumentato, in quanto il gestore di eventi timer viene chiamato più spesso. Quando si lavora in modalità tempo reale, gli eventi timer vengono generati non più di 1 volta in 10-16 millisecondi a causa di limitazioni hardware.

È possibile avviare un solo timer per ciascun programma. Ogni applicazione e chart mql5 ha una propria coda di eventi in cui vengono posizionati tutti gli eventi appena arrivati. Se la coda contiene già l'evento Timer o questo evento è in fase di elaborazione, allora il nuovo evento Timer non viene aggiunto alla coda dell'applicazione mql5.

Esempio EA con il gestore OnTimer()

```
//+------------------------------------------------------------------+
//|                                                OnTimer_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Esempio di utilizzo del timer per il calcolo dell'orario del se
#property description "Si consiglia di eseguire l'EA alla fine di una settimana di tra
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
```

```
//--- crea un timer con un periodo di 1 secondo
   EventSetTimer(1);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di deinitializzazione dell' Expert                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- distrugge il timer dopo aver completato il lavoro
   EventKillTimer();

  }
//+------------------------------------------------------------------+
//| Funzione tick Expert                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Funzione del timer                                               |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- orario della prima chiamata di OnTimer()
   static datetime start_time=TimeCurrent();
//--- orario del trade server durante la prima chiamata OnTimer()
   static datetime start_tradeserver_time=0;
//--- orario del trade server calcolato
   static datetime calculated_server_time=0;
//--- ora locale PC
   datetime local_time=TimeLocal();
//--- orario stimato del trade server
   datetime trade_server_time=TimeTradeServer();
// --- se l'orario del server è sconosciuto per qualche motivo, uscire primatempo
   if(trade_server_time==0)
      return;
//--- se il valore del trade server iniziale non è ancora impostato
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- imposta un valore calcolato del trade server
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
```

```
   else
     {
      // --- aumenta il tempo della prima chiamata di OnTimer()
      if(start_tradeserver_time!=0)
         calculated_server_time=calculated_server_time+1;;
     }
//---
   string com=StringFormat("                  Start time: %s\r\n",TimeToString(start_t
   com=com+StringFormat("                  Local time: %s\r\n",TimeToString(local_time
   com=com+StringFormat("TimeTradeServer time: %s\r\n",TimeToString(trade_server_time,
   com=com+StringFormat(" EstimatedServer time: %s\r\n",TimeToString(calculated_server
//--- visualizza i valori di tutti i contatori sul chart
   Comment(com);
  }
```

Guarda anche

EventSetTimer, EventSetMillisecondTimer, EventKillTimer, GetTickCount, GetMicrosecondCount, Eventi del terminale client

OnTrade

La funzione è chiamata nell' EA quando si verifica l'evento Trade. La funzione è pensata per elaborare le modifiche nell'ordine, posizione e liste di trade.

```
void  OnTrade(void);
```

Valore di ritorno

Nessun valore di ritorno

Nota

OnTrade() è chiamato solo per gli Expert Advisor. Non è utilizzato in indicatori e script anche se si aggiunge una funzione con lo stesso nome e tipo.

Per qualsiasi azione di trade (piazzando un ordine pendente, aprendo/chiudendo una posizione, piazzando stops, attivando ordine pendente, ecc.), la cronistoria degli ordini e delle operazioni e/o l'elenco delle posizioni e degli ordini correnti viene modificata in modo appropriato.

Quando si gestisce un ordine, il trade server invia al terminale un messaggio sull'arrivo dell'evento Trade. Per recuperare dati rilevanti su ordini e trades dalla cronistoria, è necessario eseguire una richiesta di cronologia di trading utilizzando prima HistorySelect().

Gli eventi di trade sono generati dal server in caso di:

- cambio ordini attivi,
- cambio posizione,
- cambio dels(affari),
- cambio cronistoria di trade.

Ogni evento Trade può apparire come risultato di una o più richieste di trade. Le richieste di trade vengono inviate al server utilizzando OrderSend() o OrderSendAsync(). Ogni richiesta può portare a diversi eventi di trade. Non è possibile fare affidamento sull'affermazione "Una richiesta - un evento di Trade", poiché l'elaborazione degli eventi può essere eseguita in più fasi e ogni operazione può modificare lo stato degli ordini, delle posizioni e della cronologia di trade.

L'handler OnTrade() viene chiamato dopo l'appropriata chiamata OnTradeTransaction(). In generale, non esiste una correlazione esatta nel numero di chiamate OnTrade() e OnTradeTransaction(). Una chiamata OnTrade() corrisponde a una o più chiamate OnTradeTransaction.

Esempio EA con il gestore OnTrade()

```
//+------------------------------------------------------------------+
//|                                                OnTrade_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

input    int days=7;            // profondità della cronistoria trade in giorni
```

```
//--- imposta i limiti della cronistoria trade nell'ambito globale
datetime     start;             // data di inizio per la cronistoria di trade nella ca
datetime     end;               // data di fine per la cronistoria di trade nella cach
//--- contatori globali
int          orders;            // numero di ordini attivi
int          positions;         // numero di posizioni aperte
int          deals;             // numero di affari nella cache della cronistoria di t
int          history_orders;    // numero di ordini nella cache della cronistoria di t
bool         started=false;     // flag di rilevanza contatore

//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("Limiti della cronistoria da caricare: inizio -%s, fine - %s",
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| inizializzazione di contatori di posizione, ordine e trade       |
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- carica cronistoria
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Fallimento nel caricare la cronistoria da %s a %s nella cache.
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- ottiene il valore corrente
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("Contatori di ordini, posizioni e affari inizializzati con successo");
  }
//+------------------------------------------------------------------+
//| Funzione tick Expert                                             |
//+------------------------------------------------------------------+
```

```
void OnTick()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| chiamata quando arriva un evento Trade                           |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| esempio di elaborazione delle modifiche nel trade e nella cronistoria
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- scarica la cronistoria di trading dall'intervallo specificato, nella cache del p
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Fallimento nel caricare la cronistoria da %s a %s nella cache.
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- ottiene i valori correnti
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- controlla se il numero di ordini attivi è stato modificato
   if(curr_orders!=orders)
     {
      //--- il numero di ordini attivi è stato modificato
      PrintFormat("Numero di ordini è stato cambiato. Il valore precedente è %d, il va
                  orders,curr_orders);
      //--- aggiorna il valore
      orders=curr_orders;
     }
//--- cambia il numero di posizioni aperte
   if(curr_positions!=positions)
     {
      //--- il numero di posizioni aperte è stato cambiato
      PrintFormat("Il numero di posizioni è stato cambiato Il valore precedente è %d,
                  positions,curr_positions);
      //--- aggiorna il valore
```

```
      positions=curr_positions;
     }
//--- cambia il numero di affari nella cache della cronistoria di trade
   if(curr_deals!=deals)
     {
      //--- il numero di affari nella cache della cronologia di trade è stato modifica
      PrintFormat("Il numero di affari è stato cambiato. Il valore precedente è %d, il
                  deals,curr_deals);
      //--- aggiorna il valore
      deals=curr_deals;
     }
//--- cambio nel numero degli ordini storici nella cache della cronistoria di trade
   if(curr_history_orders!=history_orders)
     {
      //--- il numero di ordini dello storico nella cache della cronistoria di trade c
      PrintFormat("Il numero di ordini nella cronistoria è stato cambiato. Il valore p
                  history_orders,curr_history_orders);
     //--- aggiornamento valore
     history_orders=curr_history_orders;
     }
//--- verifica se è necessario modificare i limiti della cronistoria di trade da richi
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//| modifica della data di inizio per la richiesta della cronistoria di trade
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
/ --- intervallo iniziale, se dovessimo iniziare a lavorare in questo momento
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- assicurarsi che il limite iniziale della cronistoria di trade non sia andato
//--- più di 1 giorno oltre la data prevista
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- corregge la data di inizio della cronistoria da caricare nella cache
      start=curr_start;
      PrintFormat("Nuovo limite iniziale della cronistoria di trade da caricare: start
                  TimeToString(start));
      //--- ora ricarica la cronistoria degli scambi per l'intervallo aggiornato
      HistorySelect(start,end);
      //--- correggi l'ordine e i contatori dell'ordine nella cronistoria per ulterio
      history_orders=HistoryOrdersTotal();
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
/* Esempio di Output :
  Limiti della cronistoria da caricare: inizio - 2018.07.16 18:11, fine 2018.07.23 18:
  I contatori di ordini, posizioni ed affari sono inizializzati con successo
```

```
   Il numero di ordini è stato cambiato. Valore precedente 0, valore corrente 1
   Il numero di ordini è stato cambiato. Valore precedente 1, valore corrente 0
   Il numero di posizioni è stato cambiato. Valore precedente 0, valore corrente 1
   Il numero di offerte è stato cambiato. Valore precedente 0, valore corrente 1
   Il numero di ordini nella cronistoria è stato modificato. Valore precedente 0, valo
*/
```

Guarda anche

OrderSend, OrderSendAsync, OnTradeTransaction, Eventi del terminale client

OnTradeTransaction

La funzione è chiamata nell' EA quando si verifica l'evento TradeTransaction. La funzione è destinata alla gestione dei risultati di esecuzione delle richieste di trade.

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // struttura transazione di trade
   const MqlTradeRequest&        request,   // struttura request (richiesta)
   const MqlTradeResult&         result     // struttura response (responso)
   );
```

Parametri

trans

[in] MqlTradeTransaction tipo di variabile che descrive una transazione effettuata su un conto di trading.

request

[in] MqlTradeRequest tipo di variabile che descrive una richiesta di trade che ha portato alla transazione. Contiene i valori per i tipi di transazione TRADE_TRANSACTION_REQUEST solo.

result

[in] MqlTradeResult tipo di variabile contenente un risultato di esecuzione di una richiesta di trade che ha portato alla transazione. Contiene i valori per i tipi di transazione TRADE_TRANSACTION_REQUEST solo.

Valore di ritorno

Nessun valore di ritorno

Nota

OnTradeTransaction() è chiamato a gestire l'evento TradeTransaction inviato dal trade server al terminale nei seguenti casi:

- invia una richiesta di trade da un programma MQL5 usando le funzioni OrderSend()/OrderSendAsync() e la sua successiva esecuzione;
- inviare manualmente una richiesta di trade tramite la GUI e la sua successiva esecuzione;
- attivazioni di ordini pendenti e stop, sul server;
- eseguire operazioni sul lato del trade server.

I dati sul tipo di transazione sono contenuti nel *tipo* di campo della variabile *trans*. I tipi di transazioni di trade sono descritti nell'enumerazione ENUM_TRADE_TRANSACTION_TYPE:

- TRADE_TRANSACTION_ORDER_ADD - aggiunta di un nuovo ordine attivo
- TRADE_TRANSACTION_ORDER_UPDATE - modifica di un ordine esistente
- TRADE_TRANSACTION_ORDER_DELETE - eliminazione di un ordine dall'elenco di quelli attivi
- TRADE_TRANSACTION_DEAL_ADD: aggiunta di un affare alla cronistoria
- TRADE_TRANSACTION_DEAL_UPDATE - modifica di un affare nella cronistoria
- TRADE_TRANSACTION_DEAL_DELETE: eliminazione di un affare dalla cronistoria
- TRADE_TRANSACTION_HISTORY_ADD - aggiunta di un ordine alla cronologia come risultato di esecuzione o cancellazione
- TRADE_TRANSACTION_HISTORY_UPDATE - modifica di un ordine nella cronistoria degli ordini

- TRADE_TRANSACTION_HISTORY_DELETE - eliminazione di un ordine dalla cronistoria degli ordini
- TRADE_TRANSACTION_POSITION - modifica della posizione non correlata ad un'esecuzione di trade
- TRADE_TRANSACTION_REQUEST - notifica che una richiesta di trade è stata elaborata dal server ed il risultato del suo trattamento è stato ricevuto.

Quando si gestiscono le transazioni del tipo TRADE_TRANSACTION_REQUEST, è necessario analizzare il secondo e il terzo parametro della funzione OnTradeTransaction() - *request* e *result* - per ricevere ulteriori informazioni.

L'invio di una richiesta di trade porta ad una catena di transazioni di trade su un conto di trading: 1) la richiesta è accettata per l'elaborazione, 2) viene creato un ordine di acquisto appropriato per l'account, 3) l'ordine viene quindi eseguito, 4) l'ordine eseguito viene rimosso dalla lista di quelli attivi, 5) aggiunto alla cronistoria degli ordini, 6) la transazione successiva viene aggiunta alla cronistoria e 7) viene creata una nuova posizione. Tutte queste fasi sono transazioni di trade. L'arrivo di ciascuna di tali transazioni al terminale è l'evento TradeTransaction. La priorità dell'arrivo di queste transazioni al terminale non è garantita. Pertanto, non dovresti aspettarti che un gruppo di transazioni arrivi dopo un altro quando sviluppi il tuo algoritmo di trading.

Quando le transazioni vengono elaborate dal gestore OnTradeTransaction() di EA, il terminale continua a gestire le transazioni di trade in entrata. Pertanto, lo stato dell'account di trading può cambiare nel corso dell'operazione OnTradeTransaction(). Ad esempio, mentre un programma MQL5 gestisce l'aggiunta di un nuovo ordine, può essere eseguito, eliminato dall'elenco degli ordini aperti e spostato nella cronistoria. Il programma viene informato di tutti questi eventi.

La lunghezza della coda delle transazioni comprende 1024 elementi. Se OnTradeTransaction() gestisce ancora un'altra transazione troppo a lungo, le precedenti possono essere sostituite da nuove transazioni nella coda.

L'handler OnTrade() viene chiamato dopo le chiamate OnTradeTransaction() appropriate. In generale, non esiste una correlazione esatta nel numero di chiamate OnTrade() e OnTradeTransaction(). Una chiamata OnTrade () corrisponde a una o più chiamate OnTradeTransaction.

Ogni evento Trade può apparire come risultato di una o più richieste di trade. Le richieste di trade vengono inviate al server utilizzando OrderSend() o OrderSendAsync(). Ogni richiesta può portare a diversi eventi di trade. Non è possibile fare affidamento sull'affermazione "Una richiesta - un evento di trade", poiché l'elaborazione degli eventi può essere eseguita in più fasi ed ogni operazione può modificare lo stato degli ordini, delle posizioni e della cronistoria di trade.

Esempio di EA con il gestore OnTradeTransaction()

```
//+------------------------------------------------------------------+
//|                                  OnTradeTransaction_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Listener di esempio di eventi TradeTransaction"
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
```

```
int OnInit()
  {
//---
   PrintFormat("LAST PING=%.f ms",
               TerminalInfoInteger(TERMINAL_PING_LAST)/1000.);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione tick Expert                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Funzione TradeTransaction                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//---
   static int counter=0;   // contatore di chiamate OnTradeTransaction()
   static uint lasttime=0; // orario dell'ultima chiamata OnTradeTransaction()
//---
   uint time=GetTickCount();
//--- se l'ultima transazione è stata eseguita più di 1 secondo fa,
   if(time-lasttime>1000)
     {
      counter=0; // allora questa è una nuova operazione di trade, il contatore può es
      if(IS_DEBUG_MODE)
         Print(" Nuova operazione di trade");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- risultato dell'esecuzione della richiesta di trade
   ulong            lastOrderID   =trans.order;
   ENUM_ORDER_TYPE  lastOrderType =trans.order_type;
   ENUM_ORDER_STATE lastOrderState=trans.order_state;
//--- il nome del simbolo, per il quale è stata eseguita una transazione
   string trans_symbol=trans.symbol;
//--- tipo di transazione
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
   switch(trans.type)
     {
      case  TRADE_TRANSACTION_POSITION:   // modifica della posizione
```

```
         {
          ulong pos_ID=trans.position;
          PrintFormat("MqlTradeTransaction: Posizione  #%d %s modificata: SL=%.5f TP=%.
                      pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
         }
       break;
       case TRADE_TRANSACTION_REQUEST:     // invio di una richiesta di trade
          PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
          break;
       case TRADE_TRANSACTION_DEAL_ADD:    // aggiunta trade
         {
          ulong           lastDealID   =trans.deal;
          ENUM_DEAL_TYPE lastDealType =trans.deal_type;
          double         lastDealVolume=trans.volume;
          // --- Trade ID in un sistema interno - un ticket assegnato da uno scambio
          string Exchange_ticket="";
          if(HistoryDealSelect(lastDealID))
             Exchange_ticket=HistoryDealGetString(lastDealID,DEAL_EXTERNAL_ID);
          if(Exchange_ticket!="")
             Exchange_ticket=StringFormat("(Exchange deal=%s)",Exchange_ticket);

          PrintFormat("MqlTradeTransaction: %s deal #%d %s %s %.2f lotti   %s",EnumToSt
                      lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolume
         }
       break;
       case TRADE_TRANSACTION_HISTORY_ADD: // aggiunta di un ordine alla cronistoria
         {
// --- ID ordine in un sistema interno - un ticket assegnato da uno scambio
          string Exchange_ticket="";
          if(lastOrderState==ORDER_STATE_FILLED)
            {
             if(HistoryOrderSelect(lastOrderID))
                Exchange_ticket=HistoryOrderGetString(lastOrderID,ORDER_EXTERNAL_ID);
             if(Exchange_ticket!="")
                Exchange_ticket=StringFormat("(Exchange ticket=%s)",Exchange_ticket);
            }
          PrintFormat("MqlTradeTransaction: %s ordine #%d %s %s %s   %s",EnumToString(t
                      lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumToStrinq
         }
       break;
       default: // altre transazioni
         {
          // --- ID ordine in un sistema interno - un ticket assegnato da Exchange
          string Exchange_ticket="";
          if(lastOrderState==ORDER_STATE_PLACED)
            {
             if(OrderSelect(lastOrderID))
                Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
             if(Exchange_ticket!="")
```

```
                Exchange_ticket=StringFormat("Exchange ticket=%s",Exchange_ticket);
            }
          PrintFormat("MqlTradeTransaction: %s ordine #%d %s %s   %s",EnumToString(tran
                      lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderSta
        }
       break;
      }
//--- ticket dell'ordine
   ulong orderID_result=result.order;
   string retcode_result=GetRetcodeID(result.retcode);
   if(orderID_result!=0)
      PrintFormat("MqlTradeResult: ordine #%d retcode=%s ",orderID_result,retcode_resu
//---
  }
//+------------------------------------------------------------------+
//| convertire i codici di risposta numerici nella stringa mnemonica |
//+------------------------------------------------------------------+
string GetRetcodeID(int retcode)
  {
   switch(retcode)
     {
      case 10004: return("TRADE_RETCODE_REQUOTE");             break;
      case 10006: return("TRADE_RETCODE_REJECT");              break;
      case 10007: return("TRADE_RETCODE_CANCEL");              break;
      case 10008: return("TRADE_RETCODE_PLACED");              break;
      case 10009: return("TRADE_RETCODE_DONE");                break;
      case 10010: return("TRADE_RETCODE_DONE_PARTIAL");        break;
      case 10011: return("TRADE_RETCODE_ERROR");               break;
      case 10012: return("TRADE_RETCODE_TIMEOUT");             break;
      case 10013: return("TRADE_RETCODE_INVALID");             break;
      case 10014: return("TRADE_RETCODE_INVALID_VOLUME");      break;
      case 10015: return("TRADE_RETCODE_INVALID_PRICE");       break;
      case 10016: return("TRADE_RETCODE_INVALID_STOPS");       break;
      case 10017: return("TRADE_RETCODE_TRADE_DISABLED");      break;
      case 10018: return("TRADE_RETCODE_MARKET_CLOSED");       break;
      case 10019: return("TRADE_RETCODE_NO_MONEY");            break;
      case 10020: return("TRADE_RETCODE_PRICE_CHANGED");       break;
      case 10021: return("TRADE_RETCODE_PRICE_OFF");           break;
      case 10022: return("TRADE_RETCODE_INVALID_EXPIRATION");  break;
      case 10023: return("TRADE_RETCODE_ORDER_CHANGED");       break;
      case 10024: return("TRADE_RETCODE_TOO_MANY_REQUESTS");   break;
      case 10025: return("TRADE_RETCODE_NO_CHANGES");          break;
      case 10026: return("TRADE_RETCODE_SERVER_DISABLES_AT");  break;
      case 10027: return("TRADE_RETCODE_CLIENT_DISABLES_AT");  break;
      case 10028: return("TRADE_RETCODE_LOCKED");              break;
      case 10029: return("TRADE_RETCODE_FROZEN");              break;
      case 10030: return("TRADE_RETCODE_INVALID_FILL");        break;
      case 10031: return("TRADE_RETCODE_CONNECTION");          break;
      case 10032: return("TRADE_RETCODE_ONLY_REAL");           break;
```

```
      case 10033: return("TRADE_RETCODE_LIMIT_ORDERS");         break;
      case 10034: return("TRADE_RETCODE_LIMIT_VOLUME");         break;
      case 10035: return("TRADE_RETCODE_INVALID_ORDER");        break;
      case 10036: return("TRADE_RETCODE_POSITION_CLOSED");      break;
      default:
         return("TRADE_RETCODE_UNKNOWN="+IntegerToString(retcode));
         break;
     }
//---
  }
```

Guarda anche

OrderSend, OrderSendAsync, OnTradeTransaction, Struttura della richiesta di Trade, Struttura delle transazioni di trade, Tipi di transazione di trade, Tipi di operazioni di trade, Eventi del terminale client

OnBookEvent

La funzione è chiamata in indicatori ed EA quando si verifica l'evento BookEvent. È pensato per gestire i cambiamenti del Depth of Market.

```
void  OnBookEvent(
   const string&  symbol         // simbolo
   );
```

Parametri

symbol

[in] Nome del simbolo per cui è arrivato BookEvent

Valore di ritorno

Nessun valore di ritorno

Nota

Per ottenere gli eventi BookEvent per qualsiasi simbolo, è sufficiente iscriversi per riceverli per quel simbolo utilizzando la funzione MarketBookAdd(). Per annullare l'iscrizione per la ricezione di BookEvent per un determinato simbolo, chiamare la funzione MarketBookRelease().

Le trasmissioni di BookEvent all'interno dell'intero grafico-chart. Ciò significa che se un'applicazione su un grafico-chart si iscrive a BookEvent utilizzando la funzione MarketBookAdd, tutti gli altri indicatori ed EA lanciati sullo stesso chart e che hanno il gestore OnBookEvent() ricevono anche questo evento. Pertanto, è necessario analizzare ilnome di simbolo passato al gestore di OnBookEvent() come parametro *symbol*.

Contatori separati di BookEvent ordinati per simboli vengono forniti per tutte le applicazioni in esecuzione sullo stesso grafico-chart. Ciò significa che ogni chart può avere più iscrizioni a simboli diversi e viene fornito un contatore per ciascun simbolo. La sottoscrizione e l'annullamento dell'iscrizione da BookEvent modifica il contatore dell'iscrizione per i simboli specificati solo all'interno di un grafico-chart. In altre parole, potrebbero esserci due chart adiacenti a BookEvent per lo stesso simbolo ma diversi valori del contatore di iscrizioni.

Il valore del contatore dell'iscrizione iniziale è zero. A ciascuna chiamata MarketBookAdd(), il contatore dell'iscrizione per un simbolo specificato sul grafico-chart è aumentato di uno (il simbolo del chart e il simbolo in MarketBookAdd() non devono corrispondere). Quando si chiama MarketBookRelease(), il contatore delle iscrizioni per un simbolo specificato all'interno del grafico-chart è diminuito di uno. Gli eventi BookEvent per qualsiasi simbolo vengono trasmessi all'interno del chart finché il contatore non è uguale a zero. Pertanto, è importante che ogni programma MQL5 che contiene MarketBookAdd() chiami correttamente l'annullamento dell'iscrizione dal ricevere eventi per ogni simbolo che utilizza MarketBookRelease() alla fine del suo lavoro. Per raggiungere questo, il numero di chiamate MarketBookAdd() e MarketBookRelease() dovrebbe essere pari per ogni chiamata durante l'intera vita del programma MQL5. L'uso di flag o contatori di iscrizioni personalizzate all'interno del programma consente di lavorare in sicurezza con gli eventi BookEvent ed impedisce di disattivare le iscrizioni per ottenere questo evento in programmi di terze parti all'interno dello stesso grafico-chart.

Gli eventi BookEvent non vengono mai saltati e vengono sempre inseriti in una coda anche se la gestione del precedente handling di BookEvent non è ancora finita.

Esempio

```
//+------------------------------------------------------------------+
//|                                          OnBookEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com/en/articles/2635"
#property version   "1.00"
#property description "Esempio di misurazione della frequenza di aggiornamento profond
#property description "Il codice è tratto dall'articolo https://www.mql5.com/en/artic
//--- parametri di input
input ulong ExtCollectTime   =30;  // tempo di test in secondi
input ulong ExtSkipFirstTicks=10;  // numero di ticks saltati all'inizio
//--- contrassegno dell'iscrizione agli eventi BookEvent
bool book_subscribed=false;
//--- array per accettare richieste dal market depth
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mostra l'inizio
   Comment(StringFormat("In attesa dell'arrivo dei primi ticks %I64u",ExtSkipFirstTic
   PrintFormat("In attesa dell'arrivo dei primi ticks %I64u",ExtSkipFirstTicks);
//--- abilita la trasmissione del market depth
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: la funzione MarketBookAdd(%s) ha restituito true",__FUNCTION__,
     }
   else
      PrintFormat("%s: la funzione MarketBookAdd(%s) ha restituito false! GetLastErro
//--- inizializzazione avvenuta con successo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Deinitializzazione Expert                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- mostra il codice della ragione di deinitializzazione
   Print(__FUNCTION__,": Codice ragione di Deinizializzazione = ",reason);
//--- cancella l'iscrizione per ottenere eventi del market depth
   if(book_subscribed)
     {
      if(!MarketBookRelease(_Symbol))
```

```
         PrintFormat("%s: MarketBookRelease(%s) ha restituito false! GetLastError()=%d
      else
         book_subscribed=false;
     }
//---
  }
//+------------------------------------------------------------------+
//| BookEvent function                                               |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;              // orario d'inizio del test
   static ulong tickcounter=0;            // aggiornamento contatore del market depth
//--- lavora con eventi di market depth solo se ci iscriviamo ad essi noi stessi
   if(!book_subscribed)
      return;
//--- conta gli aggiornamenti solo per un certo simbolo
   if(symbol!=_Symbol)
      return;
//--- salta i primi tick per cancellare la coda e prepararsi
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- ricorda l'ora di inizio
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- richiesta per i dati di market depth
   MarketBookGet(symbol,book);
//--- quando fermarsi?
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// quanto tempo è passato in microsecondi dall'inizio del test?
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%I64u ticks per %.1f secondi: %.1f ticks/sec ",ticks,endtime/1000.0
      ExpertRemove();
      return;
     }
//--- visualizza i contatori nel campo dei commenti
   if(endtime>0)
      Comment(StringFormat("%I64u ticks per %.1f secondi: %.1f ticks/sec ",ticks,endti
  }
```

Guarda anche

MarketBookAdd, MarketBookRelease, MarketBookGet, OnTrade, OnTradeTransaction, OnTick, Funzioni di Event handling, Avvio Programma, Eventi terminale client

OnChartEvent

La funzione è chiamata in indicatori e EA quando si verifica l'evento ChartEvent. La funzione è pensata per gestire le modifiche del grafico-chart effettuate da un utente o un programma MQL5.

```
void  OnChartEvent()
   const int       id,       // ID evento
   const long&     lparam,   // parametro event di tipo long
   const double&   dparam,   // parametro event di tipo double
   const string&   sparam    // parametro event di tipo string
   );
```

Parametri

id

[in] ID evento dall'enumerazione ENUM_CHART_EVENT.

lparam

[in] parametro event di tipo long

dparam

[in] parametro event di tipo double

sparam

[in] parametro event di tipo string

Valore di ritorno

Nessun valore di ritorno

Nota

Esistono 11 tipi di eventi che possono essere gestiti utilizzando la funzione OnChartEvent() predefinita. 65535 ID da CHARTEVENT_CUSTOM a CHARTEVENT_CUSTOM_LAST incluso, sono forniti per eventi personalizzati. Per generare un evento personalizzato, usare la funzione EventChartCustom().

Breve descrizione dell'evento dall'enumerazione ENUM_CHART_EVENT:

- CHARTEVENT_KEYDOWN — premendo un tasto sulla tastiera quando una finestra del grafico-chart è a fuoco;
- CHARTEVENT_MOUSE_MOVE — spostamento del mouse e click dei pulsanti del mouse (se CHART_EVENT_MOUSE_MOVE=true per un chart);
- CHARTEVENT_OBJECT_CREATE — crea un oggetto grafico (se CHART_EVENT_OBJECT_CREATE=true per un chart);
- CHARTEVENT_OBJECT_CHANGE — modifica le proprietà dell'oggetto tramite la finestra di dialogo delle proprietà;
- CHARTEVENT_OBJECT_DELETE — elimina un oggetto grafico (se CHART_EVENT_OBJECT_DELETE=true per un chart);
- CHARTEVENT_CLICK — facendo clic su un chart;
- CHARTEVENT_OBJECT_CLICK — clic del mouse su un oggetto grafico appartenente a un chart;
- CHARTEVENT_OBJECT_DRAG — trascinamento di un oggetto grafico con il mouse;

- CHARTEVENT_OBJECT_ENDEDIT — terminazione della modifica del testo nella casella di immissione Modifica di un oggetto grafico (OBJ_EDIT);
- CHARTEVENT_CHART_CHANGE — cambio di un chart;
- CHARTEVENT_CUSTOM+n — ID evento personalizzato, dove n è compreso nell'intervallo tra 0 e 65535. CHARTEVENT_CUSTOM_LAST contiene l'ultimo ID evento personalizzato accettabile (CHARTEVENT_CUSTOM+65535).

Tutti i Programmi MQL5 lavorano in thread diversi dal thread principale dell'applicazione. Il thread dell'applicazione principale è responsabile della gestione di tutti i messaggi di sistema di Windows e, a sua volta, genera messaggi Windows per la propria applicazione come risultato di questa gestione. Ad esempio, lo spostamento del mouse su un chart (evento WM_MOUSE_MOVE) genera diversi messaggi di sistema per il rendering successivo della finestra dell'applicazione e invia anche messaggi interni ad experts ed indicatori lanciati sul chart. Può verificarsi una situazione, in cui il thread dell'applicazione principale non ha ancora elaborato il messaggio di sistema WM_PAINT (e quindi non ha ancora visualizzato il chart modificato), mentre un EA o un indicatore ha già ricevuto l'evento movimento del mouse. In questo caso, la proprietà del chart CHART_FIRST_VISIBLE_BAR verrà modificata solo dopo il rendering del chart.

Per ogni tipo di evento, gli input della funzione OnChartEvent() hanno determinati valori necessari per gestire tale evento. La tabella elenca eventi e valori passati tramite i parametri.

Evento	valore del parametro 'id'	valore del parametro 'lparam'	valore del parametro 'dparam'	valore del parametro 'sparam'
Evento di battitura	CHARTEVENT_KEYDOWN	codice tasto premuto	Il numero di pressioni dei tasti generati mentre si teneva premuto il tasto nello stato premuto	Valore di stringa della maschera bit, che descrive lo stato dei tasti della tastiera
Eventi del mouse (se CHART_EVENT_MOUSE_MOVE=true per un chart)	CHARTEVENT_MOUSE_MOVE	Coordinata X	Coordinata Y	Valore stringa della maschera bit, che descrive lo stato dei tasti del mouse
Evento rotellina del mouse (se CHART_EVENT_MOUSE_WHEEL= true per il chart)	CHARTEVENT_MOUSE_WHEEL	Flag di stati di tasti e pulsanti del mouse, coordinate X e Y del cursore. Vedere la descrizione nell' esempio	Il valore Delta della rotellina del mouse	—
Creazione di un oggetto grafico (se CHART_EVENT_OBJECT_CREATE	CHARTEVENT_OBJECT_CREATE	—	—	Nome di un oggetto chart creato

Evento	valore del parametro 'id'	valore del parametro 'lparam'	valore del parametro 'dparam'	valore del parametro 'sparam'
=true per un chart)				
Modifica delle proprietà dell'oggetto tramite la finestra di dialogo delle proprietà	CHARTEVENT_OBJECT_CHANGE	—	—	Nome di un oggetto grafico modificato
Rimozione di un oggetto chart (se CHART_EVENT_OBJECT_DELETE =true per un chart)	CHARTEVENT_OBJECT_DELETE	—	—	Nome di un oggetto grafico rimosso
Clic del mouse su un chart	CHARTEVENT_CLICK	Coordinata X	Coordinata Y	—
Clic del mouse su un oggetto grafico	CHARTEVENT_OBJECT_CLICK	Coordinata X	Coordinata Y	Nome di un oggetto grafico su cui si è verificato l'evento
Spostare un oggetto grafico con il mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome di un oggetto grafico spostato
Completamento di una modifica del testo nella casella di immissione "Campo di immissione" degli oggetti grafici.	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome "Campo di immissione" dell'oggetto grafico, in cui è stata completata la modifica del testo
Ridimensionamento del chart o modifica delle proprietà del chart tramite la finestra di	CHARTEVENT_CHART_CHANGE	—	—	—

Evento	valore del parametro 'id'	valore del parametro 'lparam'	valore del parametro 'dparam'	valore del parametro 'sparam'
dialogo delle proprietà				
Evento personalizzato con numero N	CHARTEVENT_CUSTOM+N	Valore definito dalla funzione EventChartCustom()	Valore definito dalla funzione EventChartCustom()	Valore definito dalla funzione EventChartCustom()

Esempio di Listener(ascoltatore) di eventi del chart:

```
//+------------------------------------------------------------------+
//|                                          OnChartEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#proprietàdescrizione"Esempio di Listener di eventi del chart e generatore di eventi p
//--- ID chiavi di servizio
#define KEY_NUMPAD_5       12
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mostra il valore costante CHARTEVENT_CUSTOM
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Lanciato l' EA ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- imposta il flag di ricezione degli eventi di creazione dell'oggetto chart
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- imposta il flag di ricezione degli eventi di rimozione degli oggetti del chart
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- abilita i messaggi di scorrimento della rotellina del mouse
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- l'aggiornamento forzato delle proprietà del chart garantisce la preparazione per
```

```
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione tick Expert                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- contatore tick per la generazione di un evento personalizzato
   static int tick_counter=0;
//--- divide i tick accumulati con questo valore
   int simple_number=113;
//---
   tick_counter++;
//--- invia un evento personalizzato se il contatore tick è multiplo di simple_number
   if(tick_counter%simple_number==0)
     {
      //--- forma un ID evento personalizzato da 0 a 65535<
      ushort custom_event_id=ushort(tick_counter%65535);
      // --- invia un evento personalizzato con il riempimento dei parametri
      EventChartCustom(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble(Symbol
      // --- aggiunge un log per analizzare i risultati dell'esempio
      Print(__FUNCTION__,": Sent a custom event ID=",custom_event_id);
     }
//---
  }
//+------------------------------------------------------------------+
//| Funzione ChartEvent                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- keypress
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Premuto KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Premuto KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Premuto KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("Premuto KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Premuto KEY_NUMLOCK_RIGHT");  break;
         case KEY_RIGHT:         Print("Premuto KEY_RIGHT");          break;
         case KEY_NUMLOCK_DOWN:  Print("Premuto KEY_NUMLOCK_DOWN");   break;
         case KEY_DOWN:          Print("Premuto KEY_DOWN");           break;
         case KEY_NUMPAD_5:      Print("Premuto KEY_NUMPAD_5");       break;
```

```
         case KEY_NUMLOCK_5:     Print("Premuto KEY_NUMLOCK_5");      break;
         default:                Print("Premuto tasto non in lista");
        }
     }
//--- tasto sinistro del mouse sul chart
   if(id==CHARTEVENT_CLICK)
      Print("Coordinate clic del mouse su un chart x = ",lparam,"  y = ",dparam);
//--- facendo clic su un oggetto grafico
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Facendo clic sul pulsante del mouse su un oggetto denominato '"+sparam+"'
//--- oggetto rimosso
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Oggetto rimosso denominato ",sparam);
//--- oggetto creato
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("Oggetto creato denominato",sparam);
// --- oggetto modificato
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("Oggetto modificato denominato ",sparam);
//--- oggetto spostato o coordinate del punto di ancoraggio cambiate
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Modifica dei punti di ancoraggio dell'oggetto denominato ",sparam);
//--- cambiato un testo nel campo di input dell'oggetto grafico Edit
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Testo cambiato nell'oggetto Edit ",sparam,"  id=",id);
//--- eventi di movimento del mouse
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- Considera lo stato dei pulsanti e della rotella del mouse per questo event
      int flg_keys = (int)(lparam>>32);          // La flag degli stati dei tasti Ctr
      int x_cursor = (int)(short)lparam;         // coordinata X in cui si è verificat
      int y_cursor = (int)(short)(lparam>>16);   // coordinata Y dove si è verificato
      int delta    = (int)dparam;                // Valore totale dello scroll del mou
      //--- handling del flag
      string str_keys="";
      if((flg_keys&0x0001)!=0)
         str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0)
         str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0)
         str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0)
         str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0)
         str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0)
         str_keys+="X1MOUSE ";
```

```
      if((flg_keys&0x0040)!=0)
         str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1)+"'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
//--- evento di ridimensionamento del chart o modifica delle proprietà del chart utili
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("Modifica della grandezza o delle proprietà del chart");
//--- evento personalizzato
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("ID evento custom=%d, lparam=%d, dparam=%G, sparam=%s",id,lparam,dpa
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse sinistra
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse destra
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse centrale
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // primo tasto X del mouse
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // secondo tasto X del mouse
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // tasto shift
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // tasto control
   return(res);
  }
```

Guarda anche

EventChartCustom, Tipi di eventi del chart, Funzioni di Event handling, Avvio Programmi, Eventi terminale client

OnTester

La funzione viene chiamata nell' Expert Advisors quando si verifica l'evento Tester per eseguire le azioni necessarie dopo il test.

```
double  OnTester(void);
```

Valore di ritorno

Valore dell'ottimizzazione del criterio personalizzato per la valutazione dei risultati del test.

Nota

La funzione OnTester() può essere utilizzata solo durante il test di EA ed è intesa principalmente per il calcolo di un valore che viene utilizzato come criterio "Custom max" quando si ottimizzano i parametri di input.

Durante l'ottimizzazione genetica, i risultati di smistamento all'interno di una generazione vengono eseguiti in ordine decrescente. Ciò significa che i risultati con il valore più alto sono considerati i migliori dal punto di vista del criterio di ottimizzazione. I valori peggiori per tale ordinamento vengono collocati alla fine e vengono successivamente scartati. Pertanto, non prendono parte alla formazione della prossima generazione.

Pertanto, la funzione OnTester() consente non solo di creare e salvare i propri report dei risultati dei test, ma anche di controllare il processo di ottimizzazione per trovare i migliori parametri della strategia di trading.

Di seguito vi è un **esempio** di calcolare l'ottimizzazione del criterio personalizzato. L'idea è di calcolare la regressione lineare del grafico del bilancio. È descritto nell'articolo >Optimizing a strategy using balance graph and comparing results with "Balance + max Sharpe Ratio" criterion".

```
//+------------------------------------------------------------------+
//|                                                OnTester_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Esempio di EA con l'handler OnTester()"
#property description "Come criterio di ottimizzazione personalizzato, "
#property description "il rapporto tra la regressione lineare del grafico di equilibri
#property description "diviso per l'errore quadratico medio di deviazione restituito"
//--- include la classe per le operazioni di trading
#include <Trade\Trade.mqh>
//--- Parametri di input EA
input double Lots                = 0.1;      // Volume
input int    Slippage            = 10;       // Slippage ammissibile
input int    MovingPeriod        = 80;       // Periodo Media Mobile
input int    MovingShift         = 6;        // Slittamento Media Mobile
//--- variabili globali
int    IndicatorHandle=0;  // handle indicatore
```

```
bool   IsHedging=false;    // flag dell'account
CTrade trade;              // per eseguiore operazioni di trade
//---
#define EA_MAGIC 18052018
//+------------------------------------------------------------------+
//| Verifica le condizioni di apertura della posizione               |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- trade solo all'inizio di una nuova barra
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates di ",_Symbol," fallito, nessuno storico");
      return;
     }
//--- volume tick
   if(rt[1].tick_volume>1)
      return;
//--- riceve valori media mobile
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer da iMA fallito, niente dati");
      return;
     }
//--- controlla la presenza del segnale
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- candela aperta più in alto ma chiusa al di sotto della media mobile
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;    // segnale buy
   else // candela aperta in basso ma chiusa al di sopra della media mobile
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// segnale sell
     }
//--- controlli aggiuntivi
   if(signal!=WRONG_VALUE)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
        {
         double price=SymbolInfoDouble(_Symbol,signal==ORDER_TYPE_SELL ? SYMBOL_BID:SY
         trade.PositionOpen(_Symbol,signal,Lots,price,0,0);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Verifica le condizioni di chiusura della posizione               |
```

```
//+------------------------------------------------------------------+
void CheckForClose(void)
  {
   MqlRates rt[2];
//--- trade solo all'inizio di una nuova barra
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates di ",_Symbol," fallito, nessuno storico");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- riceve valori media mobile
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer da iMA fallito, niente dati");
      return;
     }
//--- la posizione è già stata selezionata in precedenza utilizzando PositionSelect()
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- candela aperta più in alto ma chiusa al di sotto della media mobile - chiude una
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- candela aperta più bassa ma chiusa sopra la media mobile - chiude una posizione
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- controlli aggiuntivi
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//| Selezionare una posizione considerando il tipo di account: Netting o Hedging
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- seleziona una posizione per un account Hedging
   if(IsHedging)
     {
      uint total=PositionsTotal();
      for(uint i=0; i<total; i++)
        {
         string position_symbol=PositionGetSymbol(i);
```

```
         if(_Symbol==position_symbol && EA_MAGIC==PositionGetInteger(POSITION_MAGIC))
           {
            res=true;
            break;
           }
        }
     }
//--- seleziona una posizione per un account Netting
   else
     {
      if(!PositionSelect(_Symbol))
         return(false);
      else
         return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---controllo del Magi
     }
//--- risultato dell'esecuzione
   return(res);
  }
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- impostazione tipo di trading: Netting o Hedging
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOU
//--- inizializzazione di un oggetto per il corretto controllo della posizione
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- crea indicatore Moving Average
   IndicatorHandle=iMA(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
     {
      printf("Errore creazione indicatore iMA");
      return(INIT_FAILED);
     }
//--- ok
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione tick Expert                                             |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- se una posizione è già aperta, controllare le condizioni di chiusura
   if(SelectPosition())
      CheckForClose();
// controlla la condizione di apertura della posizione
```

```
   CheckForOpen();
//---
  }
//+------------------------------------------------------------------+
//| Funzione tester                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- valore di ottimizzazione del criterio personalizzato (più alto è, meglio è)
   double ret=0.0;
//--- ottieni risultati di trade nell'array
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- se ci sono meno di 10 operazioni, i risultati del test non danno risultati posit
   if(trades<10)
      return (0);
//--- risultato medio per trade
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- visualizza il messaggio per la modalità test singolo
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: Trades=%d, Profitto medio=%.2f",__FUNCTION__,trades,average_pl)
//--- calcola i rapporti di regressione lineare per il grafico del profitto
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- calcola l'errore della deviazione del chart dalla linea di regressione
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- calcola il rapporto tra la tendenza profitti e la deviazione standard
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- restituisce il valore di ottimizzazione del criterio personalizzato
   return(ret);
  }
//+------------------------------------------------------------------+
//| Ottieni l'array di profitti/perdite dagli affari                 |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- richiede la cronologia completa del trading
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
   uint total_deals=HistoryDealsTotal();
   volume=0;
```

```
//--- imposta la grandezza iniziale dell'array con un margine - in base al numero di a
   ArrayResize(pl_results,total_deals);
//--- contatore di affari che fissano il risultato di trading - profitti o perdite
   int counter=0;
   ulong ticket_history_deal=0;
//--- passa attraverso tutti gli affari
   for(uint i=0;i<total_deals;i++)
     {
      //--- seleziona un affare
      if((ticket_history_deal=HistoryDealGetTicket(i))>0)
        {
         ENUM_DEAL_ENTRY deal_entry  =(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket_hi
         long            deal_type   =HistoryDealGetInteger(ticket_history_deal,DEAL_T
         double          deal_profit =HistoryDealGetDouble(ticket_history_deal,DEAL_PR
         double          deal_volume =HistoryDealGetDouble(ticket_history_deal,DEAL_VO
         //--- siamo solo interessati alle operazioni di trading
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- solo gli affari che fissano i profitti/perdite
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- scrivi il risultato del trading sull'array ed aumenta il contatore c
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- imposta la grandezza finale dell'array
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//| Calculaa la regressione lineare y=a*x+b                          |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- controlla la sufficienza dei dati
   if(ArraySize(change)<3)
      return (false);
//--- crea un array chart con un accumulo
   int N=ArraySize(change);
   ArrayResize(chartline,N);
   chartline[0]=change[0];
   for(int i=1;i<N;i++)
      chartline[i]=chartline[i-1]+change[i];
//--- ora, calcola i rapporti di regressione
   double x=0,y=0,x2=0,xy=0;
```

```
   for(int i=0;i<N;i++)
     {
      x=x+i;
      y=y+chartline[i];
      xy=xy+i*chartline[i];
      x2=x2+i*i;
     }
   a_coef=(N*xy-x*y)/(N*x2-x*x);
   b_coef=(y-a_coef*x)/N;
//---
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola l'errore di deviazione quadratica media per specificati a e b
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double &std_err)
  {
//--- somma dei quadrati di errore
   double error=0;
   int N=ArraySize(data);
   if(N<=2)
      return (false);
   for(int i=0;i<N;i++)
      error+=MathPow(a_coef*i+b_coef-data[i],2);
   std_err=MathSqrt(error/(N-2));
//---
   return (true);
  }
```

Guarda anche

Test delle strategie di trading, TesterHideIndicators, Lavorare con i risultati di ottimizzazione, TesterStatistics, OnTesterInit, OnTesterDeinit, OnTesterPass, MQL_TESTER, MQL_OPTIMIZATION, FileOpen, FileWrite, fileload, FileSave

OnTesterInit

La funzione è chiamata nell' EA quando si verifica l'evento TesterInit per eseguire le azioni necessarie prima dell'ottimizzazione nel tester di strategia. Ci sono due tipi di funzione.

La **versione che restituisce il risultato**

```
int  OnTesterInit(void);
```

Valore di ritorno

valore del tipo int, zero significa inizializzazione di successo di un EA lanciato su un chart prima dell'inizio dell'ottimizzazione.

La chiamata OnTesterInit() che restituisce il risultato dell'esecuzione è consigliata per l'uso poiché non solo consente l'inizializzazione del programma, ma restituisce anche un codice di errore in caso di una fermata di ottimizzazione anticipata. Restituzione di qualsiasi valore diverso da INIT_SUCCEEDED (0) indica un errore; non viene avviata alcuna ottimizzazione.

La versione senza un risultato restituito è lasciata solo per compatibilità con i vecchi codici. Non raccomandato per l'uso

```
void  OnTesterInit(void);
```

Nota

L'evento TesterInit viene generato prima dell'inizio dell'ottimizzazione EA nel tester strategia. A questo evento, un EA con il gestore di eventi OnTesterDeInit() o OnTesterPass() viene scaricato automaticamente su un terminale separato. Ha il simbolo e il periodo che sono stati specificati nel tester.

Tale evento riceve gli eventi TesterInit, TesterDeinit e TesterPass, ma non quelli Init, Deinit e NewTick. Di conseguenza, tutta la logica necessaria per elaborare i risultati di ogni passaggio durante l'ottimizzazione dovrebbe essere implementata negli handlers OnTesterInit(), OnTesterDeinit() e OnTesterPass() gestori.

Il risultato di ogni singolo pass durante l'ottimizzazione della strategia può essere dato tramite un frame dall handler OnTester() usando la funzione FrameAdd().

La funzione OnTesterInit() viene utilizzata per avviare un Expert Advisor prima dell'inizio dell'ottimizzazione elaborazione dei risultati di ottimizzazione. Viene sempre utilizzato insieme al gestore OnTesterDeinit().

Il tempo per l'esecuzione di OnTesterInit() è limitato. Se viene superato, l'EA viene fermato forzatamente, mentre l'ottimizzazione stessa viene annullata. Un messaggio viene visualizzato nel journal del tester:

```
TesterOnTesterInit funziona troppo a lungo. Il tester non può essere inizializzato.
```

L'esempio è preso da OnTick. Il gestore OnTesterInit() viene aggiunto per l'impostazione dei parametri di ottimizzazione:

```
//+------------------------------------------------------------------+
//|                                          OnTesterInit_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
```

```
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#proprietàdescrizione"Esempio EA con il gestore OnTesterInit()"
#property description "in cui valori e limiti di"
#property description "inputs durante l'ottimizzazione sono impostati"

input double lots=0. 1;        // volume in lots
input double kATR=3;           // signal lunghezza candela in ATR
input int    ATRperiod=20;     // ATR indicator period
input int    holdbars=8;       // numero di barre per mantenere la posizione attiva
input int    slippage=10;      // slippage consentito
input bool   revers=false;     // inverto il segnale?
input ulong  EXPERT_MAGIC=0;   // EA MagicNumber
//--- per memorizzare l'handle dell'indicatore ATR
int atr_handle;
//--- qui memorizzeremo gli ultimi valori ATR e il corpo della candela
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- ricorda l'ora di inizio dell'ottimizzazione
datetime optimization_start;
//--- per visualizzare la durata su un chart dopo la fine dell'ottimizzazione
string report;
//+------------------------------------------------------------------+
//| Funzione TesterInit                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
// --- imposta i valori degli input per l'ottimizzazione
   ParameterSetRange("lots",false,0.1,0,0,0);
   ParameterSetRange("kATR",true,3.0,1.0,0.3,7.0);
   ParameterSetRange("ATRperiod",true,10,15,1,30);
   ParameterSetRange("holdbars",true,5,3,1,15);
   ParameterSetRange("slippage",false,10,0,0,0);
   ParameterSetRange("revers",true,false,false,1,true);
   ParameterSetRange("EXPERT_MAGIC",false,123456,0,0,0);
   Print("I valori iniziali e le limitazioni dei parametri di ottimizzazione sono impo
//--- ricorda l'inizio dell'ottimizzazione
   optimization_start=TimeLocal();
   report=StringFormat("%s: ottimizzazione lanciata a %s",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|TIME_SECONDS
//--- mostra i messaggi sul chart e sul journal del terminale
   Print(report);
   Comment(report);
//---
  }
```

```
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- durata dell'ottimizzazione
   string log_message=StringFormat("%s: l'ottimizzazione ha richiesto %d secondi",
                                   __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Expert                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- inizializza le variabili globali
   last_atr=0;
   last_body=0;
//--- imposta il volume corretto
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- crea l'handle dell'indicatore ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: impossibile creare iATR, codice di errore %d",__FUNCTION__,GetI
      return(INIT_FAILED);
     }
//--- inizializzazione EA di successo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione tick Expert                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- segnale di trading
staticoint segnale = 0; // +1 significa un segnale di acquisto, -1 indica un segnale d
//--- controlla e chiude le vecchie posizioni aperte più di barre 'holdbars' fa
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- controlla una nuova barra
   if(isNewBar())
     {
      // --- controlla la presenza del segnale
      signal=CheckSignal();
     }
// --- se si apre una posizione netting, saltare il segnale - attendere fino a quando
```

```
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // esce dal gestore eventi NewTick e non entra nel mercato prima che app
     }
//--- per un conto hedging, ogni posizione viene tenuta e chiusa separatamente
   if(signal!=0)
     {
      //--- segnale buy
      if(signal>0)
        {
         PrintFormat("%s: Segnale Buy! Revers=%s",__FUNCTION__,string(revers));
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- segnale sell
      if(signal<0)
        {
         PrintFormat("%s: segnale Sell! Revers=%s",__FUNCTION__,string(revers));
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- Fine funzione OnTick
  }
//+------------------------------------------------------------------+
//| Controlla un nuovo segnale di trading                            |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 significa nessun segnale
   int res=0;
//--- ottiene il valore ATR su una penultima barra completa (l'indice della barra è 2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      // --- recupera i dati sull'ultima barra chiusa sull'array di tipo MqlRates
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         // --- calcola la misura del corpo della barra sull'ultima barra completa
         last_body=bar[0].close-bar[0].open;
// --- se il corpo dell'ultima barra (con indice 1) supera il precedente valore ATR (s
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // valore positivo per la candela verso l'alto
        }
      else
         PrintFormat("% s: impossibile ricevere l'ultima barra! Error",__FUNCTION__,Ge
```

```
      }
   else
      PrintFormat("% s: impossibile ricevere il valore dell'indicatore ATR! Error",__F
//--- se la modalità di trading inverso è abilitata
   res=revers?-res:res;  // inverte il segnale se necessario (restituisci -1 invece di
//--- restituisce un valore del segnale di trading
   return (res);
  }
//+------------------------------------------------------------------+
//| Restituisce 'true' quando appare una nuova barra                 |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // memorizza l'orario di apertura della barra corrente
//--- ottiene l'orario di apertura della barra zero
   datetime currbar_time=iTime(_Symbol,_Period,0);
// --- se l'orario di apertura cambia, è arrivata una nuova barra
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- visualizza i dati sull'orario di apertura di una nuova barra nel log
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- visualizza un messaggio con una nuova barra di apertura
         PrintFormat("%s: nuova barra su %s %s aperta a %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- recupera i dati sull'ultimo tick
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() fallito, errore = ",GetLastError());
         //--- mostra l'orario dell'ultimo tick fino ai millisecondi
         PrintFormat("L'ultimo tick era alle %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- abbiamo una nuova barra
      return (true);
     }
//--- nessuna nuova barra
   return (false);
  }
//+------------------------------------------------------------------+
//| Acquista ad un prezzo di mercato con un volume specificato       |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- compra a prezzo di mercato
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
```

```
  }
//+------------------------------------------------------------------+
//| Vendi ad un prezzo di mercato con un volume specificato          |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vendi a prezzo di mercato
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Chiusura le posizioni per hold time in barre                     |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // numero di posizioni aperte
//--- itera su posizioni aperte
   for(int i=total-1; i>=0; i--)
     {
      //--- parametri dell posizione
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- se la durata(lifetime) di una posizione è già grande, mentre MagicNumber e
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // abbassa il case del testo per la corretta formatt
         PrintFormat("Chiusura posizione #%d %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- imposta un tipo di ordine e invia una richiesta di trade
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Preparare e inviare una richiesta di trade                       |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulonq
  {
//--- dichiarare e inizializzare le strutture
```

```
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- richiesta parametri
   request.action   =TRADE_ACTION_DEAL;                     // tipi di operazioni di t
   request.position =pos_ticket;                            // ticket della posizione,
   request.symbol   =Symbol();                              // symbol
   request.volume   =volume;                                // volume
   request.type     =type;                                  // tipo di ordine
   request.price    =price;                                 // prezzo di trade
   request.deviation=slip;                                  // deviazione ammissibile
   request.magic    =magicnumber;                           // MagicNumber dell'ordine
//--- invia richiesta
   if(!OrderSend(request,result))
     {
      //--- mostra dati sul fallimento
      PrintFormat("OrderSend %s %s %.2f at %.5f errore %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- informa di un'operazione riuscita
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Guarda anche

Testing delle strategie di trading, Lavorare con i risultati di ottimizzazione, OnTesterDeinit, OnTesterPass, ParameterGetRange, ParameterSetRange

OnTesterDeinit

La funzione è chiamata in EA quando si verifica l'evento TesterDeinit dopo l'ottimizzazione EA.

```
void  OnTesterDeinit(void);
```

Valore di ritorno

Nessun valore di ritorno

Nota

L'evento TesterDeinit viene generato dopo la fine dell'ottimizzazione EA nel tester di strategia.

Un EA con un gestore di eventi OnTesterDeInit() o OnTesterPass() viene scaricato automaticamente su un terminale separato durante l'avvio dell'ottimizzazione. Ha il simbolo e il periodo che sono stati specificati nel tester. La funzione è progettata per l'elaborazione finale di tutti i risultati di ottimizzazione.

Tienere a mente che i frame di ottimizzazione inviati dagli agenti di test che utilizzano la funzione FrameAdd() possono venire in bundles e richiedere tempo per la consegna. Pertanto, non tutti i frame, così come eventi TesterPass, possono arrivare ed essere elaborati in OnTesterPass() prima della fine dell'ottimizzazione. Se si desidera ricevere tutti i frame in ritardo in OnTesterDeinit(), inserire il blocco di codice utilizzando la funzione FrameNext().

Guarda anche

Test delle strategie di trading, Lavorare con i risultati di ottimizzazione, TesterStatistics, OnTesterInit, OnTesterPass, ParameterGetRange, ParameterSetRange

OnTesterPass

La funzione è chiamata in EA quando si verifica l'evento TesterPass per la gestione di un nuovo frame di dati durante l'ottimizzazione EA.

```
void  OnTesterPass(void);
```

Valore di ritorno

Nessun valore di ritorno

Nota

L'evento TesterPass viene generato automaticamente quando si riceve un frame durante l'ottimizzazione di Expert Advisor nel tester di strategia.

Un EA con un gestore di eventi OnTesterDeInit() o OnTesterPass() viene scaricato automaticamente su un terminale separato durante l'avvio dell'ottimizzazione. Ha il simbolo e il periodo che sono stati specificati nel tester. La funzione è destinata alla gestione dei frame ricevuti dagli agenti di testing durante l'ottimizzazione. Il frame contenente i risultati del test deve essere inviato da OnTester() gestore usando la funzione FrameAdd().

Tienere a mente che i frame di ottimizzazione inviati dagli agenti di test che utilizzano la funzione FrameAdd() possono venire in bundles e richiedere tempo per la consegna. Pertanto, non tutti i frame, così come eventi TesterPass, possono arrivare ed essere elaborati in OnTesterPass() prima della fine dell'ottimizzazione. Se si desidera ricevere tutti i frame in ritardo in OnTesterDeinit(), inserire il blocco di codice utilizzando la funzione FrameNext().

Dopo aver completato l'ottimizzazione di OnTesterDeinit(), è possibile ordinare nuovamente tutti i frames ricevuti utilizzando le funzioni FrameFirst()/FrameFilter e FrameNext().

Guarda anche

Test delle strategie di trading, Lavorare con i risultati di ottimizzazione, OnTesterInit, OnTesterDeinit, FrameFirst, FrameFilter, FrameNext, FrameInputs

Come ottenere Informazioni di Mercato

Queste funzioni sono designate per ricevere informazioni sullo stato di mercato.

Funzione	Azione
SymbolsTotal	Restituisce il numero di simboli disponibili (selezionati nel Market Watch o tutti)
SymbolExist	Controlla se esiste un simbolo con il nome specificato
SymbolName	Restituisce il nome di un simbolo specificato
SymbolSelect	Consente di selezionare un simbolo nella finestra di controllo del mercato o rimuove un simbolo dalla finestra
SymbolIsSynchronized	Controlla se i dati di un simbolo selezionato nel terminale sono sincronizzati con i dati relativi al trade server
SymbolInfoDouble	Restituisce il valore double del simbolo per la proprietà corrispondente
SymbolInfoInteger	Restituisce un valore di tipo integer (long, datetime, int o bool) di un simbolo specificato per la proprietà corrispondente
SymbolInfoString	Restituisce un valore del tipo di string di un simbolo specificato per la proprietà corrispondente
SymbolInfoMarginRate	Returns the margin rates depending on the order type and direction
SymbolInfoTick	Restituisce i prezzi correnti per il simbolo di cui una variabile di tipo MqlTick
SymbolInfoSessionQuote	Permette di ricevere l'ora di inizio e fine delle sessioni quotate specificate per un simbolo e giorno della settimana specifici.
SymbolInfoSessionTrade	Permette di ricevere l'ora di inizio e fine delle sessioni di trade specificate per un simbolo e giorno della settimana specifici.
MarketBookAdd	Fornisce l'apertura della Profondità di Mercato per un simbolo selezionato, e sottoscrive per ricevere le notifiche dei cambiamenti DOM
MarketBookRelease	Fornisce la chiusura della Profondità di Mercato per un simbolo selezionato, ed annulla la sottoscrizione per ricevere le notifiche dei cambiamenti DOM
MarketBookGet	Restituisce un array di strutture MqlBookInfo contenente records della Profondità di Mercato di un simbolo specifico

SymbolsTotal

Restituisce il numero dei simboli disponibili (selezionati nel Market Watch o tutti).

```
int  SymbolsTotal(
   bool  selected      // True - solo i simboli nel MarketWatch
   );
```

Parametri

selected

[in] Modalità tipo richiesta. Può essere true o false.

Valore restituito

Se il parametro 'selezionato' è true, la funzione restituisce il numero di simboli selezionati nel MarketWatch. Se il valore è false, restituisce il numero totale di tutti i simboli.

SymbolExist

Controlla se esiste un simbolo con il nome specificato.

```
bool  SymbolExist(
   const string  name,     // nome del simbolo
   bool&   is_custom       // proprietà del simbolo personalizzato
   );
```

Parametri

name

[in] Nome del simbolo.

is_custom

[out] Proprietà del simbolo personalizzato impostata in caso di esecuzione riuscita. Se true, il simbolo rilevato è un custom (simbolo personalizzato).

Valore di ritorno

Se false, il simbolo non è stato trovato tra quelli standard e quelli personalizzati.

Guarda anche

SymbolsTotal, SymbolSelect, Custom symbols

SymbolName

Restituisce il nome di un simbolo.

```
string  SymbolName(
   int   pos,           // numero nella lista
   bool  selected       // true - solo simboli nel MarketWatch
   );
```

Parametri

pos

[in] Numero d'ordine di un simbolo.

selected

[in] Modalità tipo richiesta. Se il valore è true, il simbolo è tratto dalla lista dei simboli selezionati in MarketWatch. Se il valore è false, il simbolo è tratto dalla lista generale.

Valore restituito

Valore di tipo stringa con il nome del simbolo.

SymbolSelect

Consente di selezionare un simbolo nella finestra di controllo del market o rimuove un simbolo dalla finestra.

```
bool  SymbolSelect(
   string  name,        // nome simbolo
   bool    select       // aggiunge o rimuove
   );
```

Parametri

name

[in] Nome simbolo.

seleziona

[in] Switch. If the value is false, a symbol should be removed from MarketWatch, otherwise a symbol should be selected in this window. Un simbolo non può essere rimosso se il grafico simbolo è aperto, o non ci sono posizioni aperte per questo simbolo.

Valore restituito

In caso di fallimento restituisce false.

SymbolIsSynchronized

La funzione controlla se i dati di un simbolo selezionato nel terminale sono sincronizzati con i dati sul server di trade.

```
bool  SymbolIsSynchronized(
   string  name,        // nome simbolo
   );
```

Parametri

name

[in] Nome del Simbolo.

Return value

Se i dati sono sincronizzati, restituisce 'true', in caso contrario restituisce 'false'.

Vedi anche

SymbolInfoInteger, Organizzazione Accesso Dati

SymbolInfoDouble

Restituisce la proprietà corrispondente di un simbolo specificato. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
double  SymbolInfoDouble(
   string                    name,        // simbolo
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // identificatore della proprietà
   );
```

2. Restituisce true o false a seconda che una funzione viene eseguita con successo. In caso di successo, il valore della proprietà è posto in una variabile recipiente, passato per riferimento dall' ultimo parametro.

```
bool  SymbolInfoDouble(
   string                    name,        // simbolo
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // identificatore della proprietà
   double&                   double_var   // qui assumiamo il valore della proprietà
   );
```

Parametri

name

[in] Nome simbolo.

prop_id

[in] Identificatore di una proprietà simbolo. Il valore può essere uno dei valori dell'enumerazione ENUM_SYMBOL_INFO_DOUBLE.

double_var

[out] Variabile di tipo double ricevente il valore della proprietà richiesta.

Valore restituito

Il valore di tipo double. In caso di esecuzione fallita, le informazioni circa l'errore possono essere ottenute usando la funzione GetLastError():

- 5040 - invalid string parameter for specifying a symbol name [trad. - parametri stringa non validi per specificare il nome del simbolo]
- 4301 - unknown symbol (financial instrument) [trad. - simbolo sconosciuto (strumento finanziario)]
- 4302 - symbol is not selected in "Market Watch" (not found in the list of available ones) [trad. - il simbolo non è stato selezionato nel Market Watch" (non trovato nella lista di quelli disponibili)],
- 4303 - invalid identifier of a symbol property [trad. - identificatore della proprietà del simbolo, non valido]

Nota

E' raccomandato usare SymbolInfoTick() se la funzione viene usata per ottenere le informazioni circa l'ultimo tick. Può ben essere che non una singola citazione sia già è apparsa da quando il terminale è collegato ad un conto di trading. In tal caso, il valore richiesto sarà indefinito.

Nella maggior parte dei casi, è sufficiente utilizzare la funzione SymbolInfoTick() che permette a un utente di ricevere i valori di Ask, Bid, Last, Volume e l'orario di arrivo dell'ultimo tick durante una singola chiamata.

La funzione SymbolInfoMarginRate() fornisce dati sulla quantità di margine addebitato in base al tipo di ordine ed alla direzione.

Esempio:

```
void OnTick()
  {
//--- ottiene lo spread dalle proprietà del simbolo
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d points\r\n",
                            spreadfloat?"floating":"fixed",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- ora calcoliamo da noi stessi lo spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Calculated spread = "+(string)spread_points+" points";
   Comment(comm);
  }
```

SymbolInfoInteger

Restituisce la proprietà corrispondente di un simbolo specificato. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
long  SymbolInfoInteger(
   string                     name,      // simbolo
   ENUM_SYMBOL_INFO_INTEGER   prop_id    // identificatore della proprietà

   );
```

2. Restituisce true o false a seconda che una funzione viene eseguita con successo. In caso di successo, il valore della proprietà è posto in una variabile recipiente, passato per riferimento dall' ultimo parametro.

```
bool  SymbolInfoInteger(
   string                     name,      // simbolo
   ENUM_SYMBOL_INFO_INTEGER   prop_id,   // identificatore della proprietà
   long&                      long_var   // qui assumiamo il valore della proprietà
   );
```

Parametri

name

[in] Nome simbolo.

prop_id

[in] Identificatore di una proprietà simbolo. Il valore può essere uno dei valori dell' enumerazione ENUM_SYMBOL_INFO_INTEGER.

long_var

[out] Variabile di tipo long che riceve il valore della proprietà richiesta.

Valore restituito

Il valore di tipo long. In caso di esecuzione fallita, le informazioni circa l'errore possono essere ottenute usando la funzione GetLastError():

- 5040 - invalid string parameter for specifying a symbol name [trad. - parametri stringa non validi per specificare il nome del simbolo]
- 4301 - unknown symbol (financial instrument) [trad. - simbolo sconosciuto (strumento finanziario)]
- 4302 - symbol is not selected in "Market Watch" (not found in the list of available ones) [trad. - il simbolo non è stato selezionato nel Market Watch" (non trovato nella lista di quelli disponibili)],
- 4303 - invalid identifier of a symbol property [trad. - identificatore della proprietà del simbolo, non valido]

Nota

E' raccomandato usare SymbolInfoTick() se la funzione viene usata per ottenere le informazioni circa l'ultimo tick. Può ben essere che non una singola citazione sia già è apparsa da quando il terminale è collegato ad un conto di trading. In tal caso, il valore richiesto sarà indefinito.

Nella maggior parte dei casi, è sufficiente utilizzare la funzione SymbolInfoTick() che permette a un utente di ricevere i valori di Ask, Bid, Last, Volume e l'orario di arrivo dell'ultimo tick durante una singola chiamata.

Esempio:

```
void OnTick()
  {
//--- ottiene lo spread dalle proprietà del simbolo
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d points\r\n",
                            spreadfloat?"floating":"fixed",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- ora calcoliamo da noi stessi lo spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Calculated spread = "+(string)spread_points+" points";
   Comment(comm);
  }
```

SymbolInfoString

Restituisce la proprietà corrispondente di un simbolo specificato. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
string  SymbolInfoString(
   string                    name,         // Simbolo
   ENUM_SYMBOL_INFO_STRING   prop_id       // Identificatore proprietà
   );
```

2. Restituisce true o false, a seconda se la funzione ha avuto successo. In caso di successo, il valore della proprietà è posto in una variabile segnaposto passata per riferimento nell'ultimo parametro.

```
bool  SymbolInfoString(
   string                    name,         // Simbolo
   ENUM_SYMBOL_INFO_STRING   prop_id,      // Identificatore Proprietà
   string&                   string_var    // Qui si accetta il valore della proprietà
   );
```

Parametri

name

[in] Nome simbolo.

prop_id

[in] Identificatore di una proprietà simbolo. Il valore può essere uno dei valori dell'enumerazione ENUM_SYMBOL_INFO_STRING.

string_var

[out] Variabile di tipo stringa riceve il valore della proprietà richiesta.

Valore restituito

Il valore di tipo stringa. In caso di esecuzione fallita, le informazioni circa l'errore possono essere ottenute usando la funzione GetLastError():

- 5040 - invalid string parameter for specifying a symbol name [trad. - parametri stringa non validi per specificare il nome del simbolo]
- 4301 - unknown symbol (financial instrument) [trad. - simbolo sconosciuto (strumento finanziario)]
- 4302 - symbol is not selected in "Market Watch" (not found in the list of available ones) [trad. - il simbolo non è stato selezionato nel Market Watch" (non trovato nella lista di quelli disponibili)],
- 4303 - invalid identifier of a symbol property [trad. - identificatore della proprietà del simbolo, non valido]

Nota

E' raccomandato usare SymbolInfoTick() se la funzione viene usata per ottenere le informazioni circa l'ultimo tick. Può ben essere che non una singola citazione sia già è apparsa da quando il terminale è collegato ad un conto di trading. In tal caso, il valore richiesto sarà indefinito.

Nella maggior parte dei casi, è sufficiente utilizzare la funzione SymbolInfoTick() che permette a un utente di ricevere i valori di Ask, Bid, Last, Volume e l'orario di arrivo dell'ultimo tick durante una singola chiamata.

SymbolInfoMarginRate

Returns the margin rates depending on the order type and direction.

```
bool  SymbolInfoMarginRate(
   string              name,                    // symbol name
   ENUM_ORDER_TYPE     order_type,              // order type
   double&             initial_margin_rate,     // initial margin rate
   double&             maintenance_margin_rate  // maintenance margin rate
   );
```

Parameters

name

[in] Symbol name.

order_type

[in] Order type.

initial_margin_rate

[in] A double type variable for receiving an initial margin rate. Initial margin is a security deposit for 1 lot deal in the appropriate direction. Multiplying the rate by the initial margin, we receive the amount of funds to be reserved on the account when placing an order of the specified type.

maintenance_margin_rate

[out] A double type variable for receiving a maintenance margin rate. Maintenance margin is a minimum amount for maintaining an open position of 1 lot in the appropriate direction. Multiplying the rate by the maintenance margin, we receive the amount of funds to be reserved on the account after an order of the specified type is activated.

Return Value

Returns true if request for properties is successful, otherwise false.

SymbolInfoTick

La funzione restituisce i prezzi correnti di un simbolo specificato in una variabile del tipo MqlTick.

```
bool  SymbolInfoTick(
   string    symbol,      // symbol name
   MqlTick&  tick         // riferimento ad una struttura
   );
```

Parametri

symbol

[in] Nome del Simbolo.

tick

[out] Link alla struttura del tipo MqlTick, a cui verranno apposti il prezzo ed orario correnti dell'ultimo aggiornamento di prezzo.

Valore restituito

La funzione restituisce true in caso di successo, altrimenti restituisce false.

SymbolInfoSessionQuote

Permette di ricevere l'ora di inizio e fine delle sessioni quotate specificate per un simbolo e giorno della settimana specifici.

```
bool  SymbolInfoSessionQuote(
   string            name,                // nome simbolo
   ENUM_DAY_OF_WEEK  day_of_week,         // giorno della settimana
   uint              session_index,       // indice della sessione
   datetime&         from,                // orario di inizio della sessione
   datetime&         to                   // orario di fine della sessione
   );
```

Parametri

name

[in] Nome del Simbolo.

ENUM_DAY_OF_WEEK

[in] Giorno della settimana, il valore dell' enumerazione ENUM_DAY_OF_WEEK.

uint

[in] Numero ordinale di una sessione, il cui tempo di inizio e di fine tempo si desidera ricevere. L' indicizzazione delle sessioni inizia con 0.

da

[out] Deve essere ignorato nel valore della data restituito, l'orario di inizio della sessione in secondi da 00 ore 00 minuti.

a

[out] Deve essere ignorato nel valore della data restituito, l'orario di fine della sessione in secondi da 00 ore 00 minuti.

Valore restituito

Se i dati per la sessione specificata, simbolo e giorno della settimana vengono ricevuti, restituisce true, altrimenti restituisce false.

Vedi anche

Proprietà Simbolo, TimeToStruct, Strutture di Dati

SymbolInfoSessionTrade

Permette di ricevere l'ora di inizio e fine delle sessioni di trade specificate per un simbolo e giorno della settimana specifici.

```
bool  SymbolInfoSessionTrade(
   string            name,                // nome simbolo
   ENUM_DAY_OF_WEEK  day_of_week,         // giorno della settimana
   uint              session_index,       // indice della sessione
   datetime&         from,                // orario inizio sessione
   datetime&         to                   // orario fine sessione
   );
```

Parametri

name

[in] Nome del Simbolo.

ENUM_DAY_OF_WEEK

[in] Giorno della settimana, il valore dell' enumerazione ENUM_DAY_OF_WEEK.

uint

[in] Numero ordinale di una sessione, il cui tempo di inizio e di fine tempo si desidera ricevere. L' indicizzazione delle sessioni inizia con 0.

da

[out] Deve essere ignorato nel valore della data restituito, l'orario di inizio della sessione in secondi da 00 ore 00 minuti.

a

[out] Deve essere ignorato nel valore della data restituito, l'orario di fine della sessione in secondi da 00 ore 00 minuti.

Return value

Se i dati per la sessione specificata, simbolo e giorno della settimana vengono ricevuti, restituisce true, altrimenti restituisce false.

Vedi anche

Proprietà Simbolo, TimeToStruct, Strutture di Dati

MarketBookAdd

Fornisce l'apertura della profondità di mercato per un simbolo selezionato, e sottoscrive per ricevere le notifiche dei cambiamenti DOM.

```
bool  MarketBookAdd(
   string  symbol      // simbolo
   );
```

Parametri

symbol

[In] Il nome di un simbolo, la cui profondità di mercato deve essere utilizzata nell' Expert Advisor o script.

Valore restituito

Il valore true se aperto con successo, altrimenti false.

Nota

Normalmente, questa funzione deve essere chiamata dalla funzione OnInit() o nel costruttore della classe. Per gestire gli avvisi in arrivo, nel programma Expert Advisor deve contenere la funzione void OnBookEvent(stringa e simbolo).

Vedi anche

Struttura della Profondità di Mercato, Strutture e Classi

MarketBookRelease

Fornisce la chiusura della Profondità di Mercato per un simbolo selezionato, e annulla la sottoscrizione per ricevere le notifiche dei cambiamenti DOM.

```
bool  MarketBookRelease(
   string  symbol      // simbolo
   );
```

Parametri

symbol

[in] Nome simbolo.

Valore restituito

Il valore true se chiuso con successo, altrimenti false.

Nota

Normalmente, questa funzione deve essere chiamata dalla funzione OnDeinit(), se la corrispondente funzione MarketBookAdd() è stata richiamata nella funzione OnInit(). Oppure deve essere chiamata dal distruttore della classe, se la corrispondente funzione MarketBookAdd() è stata chiamata dal costruttore della classe.

Vedi anche

Struttura della Profondità di Mercato,Strutture e Classi

MarketBookGet

Restituisce un array di strutture MqlBookInfo contenente i record della Profondità di Mercato di un simbolo specifico.

```
bool  MarketBookGet(
   string        symbol,      // simbolo
   MqlBookInfo&  book[]       // riferimento ad un array
   );
```

Parametri

symbol

[in] Nome simbolo.

book[]

[In] Riferimento a un array di records di Profondità di Mercato. L' array può essere pre-assegnato per un numero sufficiente di record. Se un array dinamico non è stato pre-assegnato nella memoria operativa, il terminale client distribuirà l'array stesso.

Valore restituito

Restituisce true in caso di successo, altrimenti false.

Nota

La profondità di mercato deve essere pre-aperto dalla funzione MarketBookAdd().

Esempio:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo per ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("Non si può ottenere il contenuto del simbolo DOM ",Symbol());
     }
```

Vedi anche

Struttura della Profondità di Mercato,Strutture e Classi

Funzioni del calendario economico

Questa sezione descrive le funzioni per lavorare con il calendario economico disponibile direttamente nella piattaforma MetaTrader. Il calendario economico è un'enciclopedia già pronta con descrizioni di indicatori macroeconomici, le loro date di rilascio e gradi di importanza. I valori rilevanti degli indicatori macroeconomici vengono inviati alla piattaforma MetaTrader proprio al momento della pubblicazione e vengono visualizzati su un chart come tag che consente di monitorare visivamente gli indicatori richiesti in base a Paesi, valute e importanza.

Tutte le funzioni per lavorare con il calendario economico utilizzano l'ora del trade server (TimeTradeServer). Ciò significa che il tempo nella struttura MqlCalendarValue e gli input di tempo nelle funzioni CalendarValueHistoryByEvent/CalendarValueHistory sono impostate nel fuso orario del trade server, piuttosto che nell'ora locale dell' utente.

Le funzioni del calendario economico consentono di condurre l'analisi automatica degli eventi in arrivo in base a criteri di importanza personalizzati da una prospettiva del paese/valuta necessario.

Funzione	Azione
CalendarCountryById	Ottiene una descrizione del Paese tramite il suo ID
CalendarEventById	Ottiene una descrizione dell'evento tramite il suo ID
CalendarValueById	Ottiene una descrizione del valore dell'evento tramite il suo ID
CalendarCountries	Ottiene la serie di nomi dei Paesi disponibili nel calendario
CalendarEventByCountry	Ottiene l'array delle descrizioni di tutti gli eventi disponibili nel calendario con un codice Paese specificato
CalendarEventByCurrency	Ottiene l'array di descrizioni di tutti gli eventi disponibili nel calendario in base ad una valuta specificata
CalendarValueHistoryByEvent	Ottiene l'array di valori per tutti gli eventi in un intervallo di tempo specificato da un ID evento
CalendarValueHistory	Ottiene l'array di valori per tutti gli eventi in un intervallo di tempo specificato con la possibilità di ordinare per Paese e/o valuta
CalendarValueLastByEvent	Ottiene l'array di valori di eventi in base al suo ID dallo stato del database del calendario con un change_id specificato
CalendarValueLast	Ottiene l'array di valori per tutti gli eventi con la possibilità di ordinare per Paese e/o valuta dallo stato del database del calendario con un change_id specificato

CalendarCountryById

Ottiene una descrizione del Paese tramite il suo ID.

```
bool  CalendarCountryById(
   const long           country_id,     // ID Paese
   MqlCalendarCountry&  country         // variabile per ricevere una descrizione del
   );
```

Parametri

country_id

[in] Country ID (ISO 3166-1).

country

[out] MqlCalendarCountry: è il tipo di variabile per ricevere una descrizione del Paese.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 5402 - ERR_CALENDAR_NO_DATA (Paese non trovato),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto).

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ottiene l'elenco dei paesi dal calendario economico
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- controlla il risultato
   if(count==0)
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
//--- se ci sono due o più Paesi
   if(count>=2)
    {
      MqlCalendarCountry country;
      //--- ora ottiene una descrizione del Paese tramite il suo ID
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- prepara una descrizione del Paese
         string descr="id = "+IntegerToString(country.id)+"\n";
         descr+=("name = " + country.name+"\n");
         descr+=("code = " + country.code+"\n");
         descr+=("currency = " + country.currency+"\n");
         descr+=("currency_symbol = " + country.currency_symbol+"\n");
```

```
         descr+=("url_name = " + country.url_name);
         //--- mostra una descrizione del Paese
         Print(descr);
        }
      else
         Print("CalendarCountryById() fallito. Error ",GetLastError());
     }
//---
  }
/*
  Risultato:
   id = 999
   name = European Union
   code = EU
   currency = EUR
   currency_symbol = €
   url_name = european-union
*/
```

Guarda anche

CalendarCountries, CalendarEventByCountry

CalendarEventById

Ottieni una descrizione dell'evento tramite il suo ID.

```
bool  CalendarEventById(
   ulong                event_id,      // ID evento
   MqlCalendarEvent&    event          // variabile per ricevere una descrizione dell'e
   );
```

Parametri

event_id

[in] Event ID.

event

[out] MqlCalendarEvent: è il tipo di variabile per ricevere una descrizione dell'evento.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 5402 - ERR_CALENDAR_NO_DATA (Paese non trovato),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto).

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- codice Paese per la Germania (ISO 3166-1 Alpha-2)
   string germany_code="DE";
//--- ottiene eventi Tedeschi
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- mostra gli eventi Tedeschi nel Journal
   if(events_count>0)
    {
      PrintFormat("Germany events: %d",events_count);
      ArrayPrint(events);
     }
   else
    {
      PrintFormat("Impossibile ricevere eventi per il codice Paese %s, errore %d",
                  germany_code,GetLastError());
//--- completamento precoce script
      return;
     }
//--- ottiene la descrizione dell'ultimo evento dall'array events[]
```

```
   MqlCalendarEvent event;
   ulong event_id=events[events_count-1].id;
   if(CalendarEventById(event_id,event))
    {
      MqlCalendarCountry country;
      CalendarCountryById(event.country_id,country);
      PrintFormat("Descrizione evento con event_id=%d ricevuto",event_id);
      PrintFormat("Paese: %s (country code = %d)",country.name,event.country_id);
      PrintFormat("Nome evento: %s",event.name);
      PrintFormat("Codice evento: %s",event.event_code);
      PrintFormat("Importanza evento: %s",EnumToString((ENUM_CALENDAR_EVENT_IMPORTANCE
      PrintFormat("Tipo di evento: %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)event.ty
      PrintFormat("Settore evento: %s",EnumToString((ENUM_CALENDAR_EVENT_SECTOR)event.
      PrintFormat("Freuenza evento: %s",EnumToString((ENUM_CALENDAR_EVENT_FREQUENCY)ev
      PrintFormat("Modalità rilascio evento: %s",EnumToString((ENUM_CALENDAR_EVENT_TIM
      PrintFormat("Unità di misura evento: %s",EnumToString((ENUM_CALENDAR_EVENT_UNIT)
      PrintFormat("Numero di cifre decimali: %d",event.digits);
      PrintFormat("Moltiplicatore evento: %s",EnumToString((ENUM_CALENDAR_EVENT_MULTIP
      PrintFormat("URL sorgente: %s",event.source_url);
    }
   else
      PrintFormat("Impossibile ottenere la descrizione dell'evento per event_d =%s, er
                  event_id,GetLastError());
  }
/*
  Risultato:
  Eventi Tedeschi: 50
             [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importa
   [ 0] 276010001      1        6           2           0          276      1
   [ 1] 276010002      1        6           2           0          276      1
   [ 2] 276010003      1        4           2           0          276      1
   [ 3] 276010004      1        4           2           0          276      1
   ....
   [47] 276500001      1        8           2           0          276      0
   [48] 276500002      1        8           2           0          276      0
   [49] 276500003      1        8           2           0          276      0
  Descrizione dell'evento con event_id = 276500003 ricevuto
  Paese: Germania (codice Paese = 276)
  Nome dell'evento: Markit PMI Composito
  Codice evento: markit-composite-pmi
   Importanza evento: CALENDAR_IMPORTANCE_MODERATE
   Tipo di evento: CALENDAR_TYPE_INDICATOR
   Settore evento: CALENDAR_SECTOR_BUSINESS
   Frequenza evento: CALENDAR_FREQUENCY_MONTH
   Modalità rilascio evento: CALENDAR_TIMEMODE_DATETIME
   Unità di misura evento: CALENDAR_UNIT_NONE
  Numero di posizioni decimali: 1
   Valore moltiplicatore: CALENDAR_MULTIPLIER_NONE
   URL sorgente: https://www.markiteconomics.com
```

```
*/
```

Guarda anche

CalendarEventByCountry, CalendarEventByCurrency, CalendarValueById

CalendarValueById

Ottiene una descrizione del valore dell'evento tramite il suo ID.

```
bool  CalendarValueById(
   ulong                value_id,      // ID valore evento
   MqlCalendarValue&    value          // variabile per la ricezione di un valore event
   );
```

Parametri

value_id

[in] ID valore evento.

value

[out] MqlCalendarValue: è il tipo di variabile per ricevere una descrizione dell'evento.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 5402 - ERR_CALENDAR_NO_DATA (Paese non trovato),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto).

Nota

Tutte le funzioni per lavorare con il calendario economico utilizzano l'ora del trade server (TimeTradeServer). Ciò significa che il tempo nella struttura MqlCalendarValue e gli input di tempo nelle funzioni CalendarValueHistoryByEvent/CalendarValueHistory sono impostate nel fuso orario del trade server, piuttosto che nell'ora locale dell' utente.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- codice Paese per il Giappone (ISO 3166-1 Alpha-2)
   string japan_code="JP";
//--- imposta i limiti dell'intervallo da cui prendiamo gli eventi
   datetime date_from=D'01.01.2018';  // prende tutti gli eventi dal 2018
   datetime date_to=0;                // 0 indica tutti gli eventi noti, inclusi quel
//--- ottiene l'arra dei valori degli eventi in Giappone
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- sposta i valori degli eventi rilevati
   if(values_count>0)
    {
      PrintFormat("Numero di valori per gli eventi in Giappone: %d",values_count);
      //--- elimina tutti i valori "vuoti" (actual_value==-9223372036854775808)
```

```
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("Numero di valori dopo aver eliminato quelli vuoti: %d",ArraySize(va
     }
   else
    {
      PrintFormat("Impossibile ricevere eventi per il codice Paese %s, errore %d",
                  japan_code,GetLastError());
//--- completamento precoce script
      return;
     }
//--- lascia non più di 10 valori nell' array values[]
   if(ArraySize(values)>10)
    {
      PrintFormat("Riduce l'elenco dei valori a 10 e li visualizza");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- ora mostriamo come ottenere una descrizione del valore di un evento in base al v
   for(int i=0;i<ArraySize(values);i++)
    {
      MqlCalendarValue value;
      CalendarValueById(values[i].id,value);
      PrintFormat("%d: value_id=%d value=%d impact=%s",
                  i,values[i].id,value.actual_value,EnumToString(ENUM_CALENDAR_EVENT_I
     }
//---
  }
/*
  Risultato:
  Numero di valori per gli eventi in Giappone: 1734
  Numero di valori dopo aver cancellato quelli vuoti: 1017
  Riduce l'elenco dei valori a 10 e li visualizza
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 56500  392030004 2019.03.28 23:30:00 2019.03.01 00:00:00          0         900
   [1] 56501  392030005 2019.03.28 23:30:00 2019.03.01 00:00:00          0         700
   [2] 56502  392030006 2019.03.28 23:30:00 2019.03.01 00:00:00          0        1100
   [3] 56544  392030007 2019.03.28 23:30:00 2019.02.01 00:00:00          0        2300
   [4] 56556  392050002 2019.03.28 23:30:00 2019.02.01 00:00:00          0        1630
   [5] 55887  392020003 2019.03.28 23:50:00 2019.02.01 00:00:00          0         400
   [6] 55888  392020004 2019.03.28 23:50:00 2019.02.01 00:00:00          0       -1800
   [7] 55889  392020002 2019.03.28 23:50:00 2019.02.01 00:00:00          0         200
   [8] 55948  392020006 2019.03.28 23:50:00 2019.02.01 00:00:00          1        1400
   [9] 55949  392020007 2019.03.28 23:50:00 2019.02.01 00:00:00          1       -1000
  Mostra dati brevi sui valori degli eventi in base a value_id
```

```
   0: value_id=56500 value=900000 impact=CALENDAR_IMPACT_POSITIVE
   1: value_id=56501 value=700000 impact=CALENDAR_IMPACT_NA
   2: value_id=56502 value=1100000 impact=CALENDAR_IMPACT_POSITIVE
   3: value_id=56544 value=2300000 impact=CALENDAR_IMPACT_NEGATIVE
   4: value_id=56556 value=1630000 impact=CALENDAR_IMPACT_POSITIVE
   5: value_id=55887 value=400000 impact=CALENDAR_IMPACT_NEGATIVE
   6: value_id=55888 value=-1800000 impact=CALENDAR_IMPACT_POSITIVE
   7: value_id=55889 value=200000 impact=CALENDAR_IMPACT_NEGATIVE
   8: value_id=55948 value=1400000 impact=CALENDAR_IMPACT_POSITIVE
   9: value_id=55949 value=-1000000 impact=CALENDAR_IMPACT_NEGATIVE
*/
```

Guarda anche

CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent, CalendarValueLast

CalendarCountries

Ottiene la matrice dei nomi dei paesi disponibili nel Calendario.

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]         // array per la ricezione di un elenco di
   );
```

Parametri

countries[]

[out] Un array di tipo MqlCalendarCountry per ricevere descrizioni di tutti i Paesi del calendario.

Valore di ritorno

Numero di descrizioni ricevute. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto),
- 5400 - ERR_CALENDAR_MORE_DATA (la dimensione dell'array non è sufficiente per ricevere le descrizioni di tutti i Paesi, solo quelli che sono riusciti a rientrare sono stati ricevuti).

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ottiene l'elenco dei paesi dal calendario economico
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- visualizza l'array nel Journal
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
/*
  Risultato:
        [id]            [name] [code] [currency] [currency_symbol]       [url_name] [re
   [ 0]    0 "Mondiale"       "WW"   "ALL"      ""                 "mondiale"
   [ 1]  999 "Unione Europea" "EU"   "EUR"      "€"                "unione-europea"
   [ 2]  840 "Stati uniti"  "US"   "USD"      "$"                "stati-uniti"
   [ 3]  124 "Canada"         "CA"   "CAD"      "$"                "canada"
   [ 4]   36 "Australia"      "AU"   "AUD"      "$"                "australia"
   [ 5]  554 "Nuova Zelanda"    "NZ"   "NZD"      "$"                "nuova-zelanda"
   [ 6]  392 "Giappone"          "JP"   "JPY"      "Ґ"                "giappone"
   [ 7]  156 "Cina"           "CN"   "CNY"      "Ґ"                "cina"
   [ 8]  826 "Regno Unito" "GB"   "GBP"      "J"                "regno-unito"
   [ 9]  756 "Svizzera"    "CH"   "CHF"      "₣"                "svizzera"
```

```
   [10]  276 "Germania"         "DE"   "EUR"      "€"                "germania"
   [11]  250 "Francia"          "FR"   "EUR"      "€"                "francia"
   [12]  380 "Italia"           "IT"   "EUR"      "€"                "italia"
   [13]  724 "Spagna"           "ES"   "EUR"      "€"                "spagna"
   [14]   76 "Brasile"          "BR"   "BRL"      "R$"               "brasile"
   [15]  410 "Sud Corea"    "KR"   "KRW"      "₩"              "sud-corea"
*/
  }
```

Guarda anche

CalendarEventByCountry, CalendarCountryById

CalendarEventByCountry

Ottiene l'array delle descrizioni di tutti gli eventi disponibili nel Calendario con un codice Paese specificato.

```
int  CalendarEventByCountry(
   string               country_code,     // nome del codice del Paese (ISO 3166-1 alp
   MqlCalendarEvent&    events[]          // variabile per ricevere l'array di descri
   );
```

Parametri

country_code

[in] Nome codice Paese (ISO 3166-1 alpha-2)

events[]

[out] MqlCalendarEvent: è il tipo di array per la ricezione di descrizioni di tutti gli eventi per un Paese specificato.

Valore di ritorno

Numero di descrizioni ricevute. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insufficiente per l'esecuzione di una richiesta),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto),
- errori di esecuzione fallita di ArrayResize()

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- codice Paese per EU (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- ottieni eventi UE
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- visualizza gli eventi UE nel Journal
   if(events_count>0)
    {
      PrintFormat("eventi EU: %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
  Risultato:
  eventi EU: 56
```

```
               [id] [type]   [country_id] [unit] [importance] [multiplier] [digits] [ever
   [ 0] 999010001      0            999      0             2            0         0  "ECB
   [ 1] 999010002      0            999      0             2            0         0  "ECB
   [ 2] 999010003      0            999      0             3            0         0  "ECB
   [ 3] 999010004      0            999      0             3            0         0  "ECB
   [ 4] 999010005      0            999      0             2            0         0  "ECB
   [ 5] 999010006      1            999      1             3            0         2  "ECB
   [ 6] 999010007      1            999      1             3            0         2  "ECB
   [ 7] 999010008      0            999      0             2            0         0  "ECB
   [ 8] 999010009      1            999      2             2            3         3  "ECB
   [ 9] 999010010      0            999      0             2            0         0  "ECB
   [10] 999010011      0            999      0             2            0         0  "ECB
   ...

*/
```

Guarda anche

CalendarCountries, CalendarCountryById

CalendarEventByCurrency

Ottieni l'array delle descrizioni di tutti gli eventi disponibili nel Calendario in base ad una valuta specificata.

```
int  CalendarEventByCountry(
   const string         currency,     // nome del codice valuta del Paese
   MqlCalendarEvent&    events[]      // variabile per ricevere l'array della descriz
   );
```

Parametri

currency

[in] Nome del codice valuta del Paese.

events[]

[out] MqlCalendarEvent: tipo di array per la ricezione delle descrizioni di tutti gli eventi per una valuta specificata.

Valore di ritorno

Numero di descrizioni ricevute. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insufficiente per l'esecuzione di una richiesta),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto),
- errori di esecuzione fallita di ArrayResize()

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dichiara l'array per ricevere gli eventi del Calendario Economico
      MqlCalendarEvent events[];
//--- ottieni eventi valutari UE
      int count = CalendarEventByCurrency("EUR",events);
      Print("count = ", count);
//--- 10 eventi sono sufficienti per l'esempio corrente
      if(count>10)
         ArrayResize(events,10);
//--- visualizza gli eventi nel diario
      ArrayPrint(events);
  }
/*
  Risultato:
             [id] [type] [country_id] [unit] [importance]
   [0] 999010001      0          999      0            2 "https://www.ecb.europa.eu/h
```

```
   [1] 999010002      0           999      0            2 "https://www.ecb.europa.eu/hc
   [2] 999010003      0           999      0            3 "https://www.ecb.europa.eu/hc
   [3] 999010004      0           999      0            3 "https://www.ecb.europa.eu/hc
   [4] 999010005      0           999      0            2 "https://www.ecb.europa.eu/hc
   [5] 999010006      1           999      1            3 "https://www.ecb.europa.eu/hc
   [6] 999010007      1           999      1            3 "https://www.ecb.europa.eu/hc
   [7] 999010008      0           999      0            2 "https://www.ecb.europa.eu/hc
   [8] 999010009      1           999      2            2 "https://www.ecb.europa.eu/hc
   [9] 999010010      0           999      0            2 "https://www.ecb.europa.eu/hc
*/
```

Guarda anche

CalendarEventById, CalendarEventByCountry

CalendarValueHistoryByEvent

Ottiene l'array di valori per tutti gli eventi in un intervallo di tempo specificato da un ID evento.

```
bool  CalendarValueById(
   ulong              event_id,          // ID evento
   MqlCalendarValue&  values[],          // array per le descrizioni dei valori
   datetime           datetime_from,     // bordo sinistro di un intervallo di tempo
   datetime           datetime_to=0      // bordo destro di un intervallo di tempo
   );
```

Parametri

event_id

[in] Event ID.

values[]

[out] MqlCalendarValue: è il tipo di array per ricevere i valori degli eventi

datetime_from

[in] La data iniziale di un intervallo temporale viene selezionata da un ID specificato, mentre *datetime_from* < *datetime_to*.

datetime_to=0

[in] La data di fine di un intervallo di tempo viene selezionata da un ID specificato. Se *datetime_to* non è impostato (o è 0), tutti i valori degli eventi che iniziano da quelli specificati dalla data *datetime_from* nel database del calendario vengono restituiti (compresi i valori degli eventi futuri).

Valore di ritorno

Restituisce true se ha successo, altrimenti - false. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insufficiente per l'esecuzione di una richiesta),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto),
- 5400 - ERR_CALENDAR_MORE_DATA (la dimensione dell'array è insufficiente per la ricezione delle descrizioni di tutti i valori, solo quelli che sono riusciti a rientrare sono stati ricevuti),
- errori di esecuzione fallita di ArrayResize()

Se un valore di un evento non ha nessuno dei campi specificati di seguito

```
struct MqlCalendarValue
  {
   ...
   long              actual_value;              // valore attuale dell'evento
   long              prev_value;                // valore precedente dell'evento
   long              revised_prev_value;        // valore rivisto dell'evento pr
   long              forecast_value;            // valore di previsione dell'eve
   ...
  };
```

allora il valore del campo mancante viene restituito come INT64_MIN (-9223372036854775808). Vedere il valore campo *revised_prev_value* nell'esempio seguente.

Nota

Tutte le funzioni per lavorare con il calendario economico utilizzano l'ora del trade server (TimeTradeServer). Ciò significa che il tempo nella struttura MqlCalendarValue e gli input di tempo nelle funzioni CalendarValueHistoryByEvent/CalendarValueHistory sono impostate nel fuso orario del trade server, piuttosto che nell'ora locale dell' utente.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- codice Paese per EU (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- ottieni eventi UE
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- visualizza gli eventi UE nel Journal
   if(events_count>0)
    {
      PrintFormat("eventi EU: %d",events_count);
      //--- riduce la lista degli eventi, 10 eventi sono sufficienti per l'analisi
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- vedi che l'evento "Decisione tasso di interesse della BCE" abbia event_id = 999
   ulong event_id=events[6].id;        // l'ID dell'evento può cambiare nel Calendario
   string event_name=events[6].name;   // nome di un evento del calendario
   PrintFormat("Ottiene valori per event_name =%s event_id=%d",event_name,event_id);
//--- ottiene tutti i valori dell'evento "Decisione tasso di interesse della BCE"
   MqlCalendarValue values[];
//--- imposta i limiti dell'intervallo da cui prendiamo gli eventi
   datetime date_from=0;           // prende tutti gli eventi dall'inizio della crono
   datetime date_to=D'01.01.2016'; // prendere eventi non più vecchi del 2016
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
    {
      PrintFormat("Valori ricevuti per %s: %d",
                  event_name,ArraySize(values));
      //--- riduci la lista valori, 10 eventi sono sufficienti per l'analisi
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
    {
      PrintFormat("Errore! Impossibile ottenere valori per event_id=% d ", Event_id);
```

```
      PrintFormat("Codice errore: %d",GetLastError());
     }
  }
//---
/*
  Risultato:
  eventi EU: 56
           [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importar
   [0] 999010001      0        5           0           0           999      0
   [1] 999010002      0        5           0           0           999      0
   [2] 999010003      0        5           0           0           999      0
   [3] 999010004      0        5           0           0           999      0
   [4] 999010005      0        5           0           0           999      0
   [5] 999010006      1        5           0           0           999      1
   [6] 999010007      1        5           0           0           999      1
   [7] 999010008      0        5           0           0           999      0
   [8] 999010009      1        5           0           0           999      2
   [9] 999010010      0        5           0           0           999      0
  Ottieni valori per event_name=Decisione tasso di interesse ECB event_id = 999010007
  Valori degli eventi della Decisione sui tassi di interesse della BCE ricevuti: 102
      [id] [event_id]              [time]            [period] [revision] [actual_valu
   [0] 2776  999010007 2007.03.08 11:45:00 1970.01.01 00:00:00          0        3750(
   [1] 2777  999010007 2007.05.10 11:45:00 1970.01.01 00:00:00          0        3750(
   [2] 2778  999010007 2007.06.06 11:45:00 1970.01.01 00:00:00          0        4000(
   [3] 2779  999010007 2007.07.05 11:45:00 1970.01.01 00:00:00          0        4000(
   [4] 2780  999010007 2007.08.02 11:45:00 1970.01.01 00:00:00          0        4000(
   [5] 2781  999010007 2007.09.06 11:45:00 1970.01.01 00:00:00          0        4000(
   [6] 2782  999010007 2007.10.04 11:45:00 1970.01.01 00:00:00          0        4000(
   [7] 2783  999010007 2007.11.08 12:45:00 1970.01.01 00:00:00          0        4000(
   [8] 2784  999010007 2007.12.06 12:45:00 1970.01.01 00:00:00          0        4000(
   [9] 2785  999010007 2008.01.10 12:45:00 1970.01.01 00:00:00          0        4000(
*/
```

Guarda anche

CalendarCountries, CalendarEventByCountry, CalendarValueHistory, CalendarEventById, CalendarValueById

CalendarValueHistory

Ottiene l'array di valori per tutti gli eventi in un intervallo di tempo specificato con la possibilità di ordinare per Paese e/o valuta.

```
bool  CalendarValueById(
   MqlCalendarValue&  values[],              // array per le descrizioni dei valori
   datetime           datetime_from,         // bordo sinistro dell' intervallo di ter
   datetime           datetime_to=0          // bordo destro dell' intervallo di tempo
   const string       country_code=NULL,     // codice nome Paese (ISO 3166-1 alpha-2)
   const string       currency=NULL          // codice nome della valuta del Paese
   );
```

Parametri

values[]

[out] MqlCalendarValue: è il tipo di array per ricevere i valori degli eventi

datetime_from

[in] La data iniziale di un intervallo temporale viene selezionata da un ID specificato, mentre *datetime_from* < *datetime_to*.

datetime_to=0

[in] La data di fine di un intervallo di tempo viene selezionata da un ID specificato. Se *datetime_to* non è impostato (o è 0), tutti i valori degli eventi che iniziano da quelli specificati dalla data *datetime_from* nel database del calendario vengono restituiti (compresi i valori degli eventi futuri).

country_code=NULL

[in] Nome codice Paese (ISO 3166-1 alpha-2)

currency=NULL

[in] Nome del codice valuta del Paese.

Valore di ritorno

Restituisce true se ha successo, altrimenti - false. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insufficiente per l'esecuzione di una richiesta),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto),
- 5400 - ERR_CALENDAR_MORE_DATA (la dimensione dell'array è insufficiente per la ricezione delle descrizioni di tutti i valori, solo quelli che sono riusciti a rientrare sono stati ricevuti),
- errori di esecuzione fallita di ArrayResize()

Nota

Tutte le funzioni per lavorare con il calendario economico utilizzano l'ora del trade server (TimeTradeServer). Ciò significa che il tempo nella struttura MqlCalendarValue e gli input di tempo nelle funzioni CalendarValueHistoryByEvent/CalendarValueHistory sono impostate nel fuso orario del trade server, piuttosto che nell'ora locale dell' utente.

Se l'array *events[]* di lunghezza fissa è stato passato alla funzione e non c'era spazio sufficiente per salvare l'intero risultato, l'errore ERR_CALENDAR_MORE_DATA (5400) viene attivato.

Se *datetime_to* non è impostato (o è 0), tutti i valori degli eventi che iniziano da quelli specificati dalla data *datetime_from* nel database del calendario vengono restituiti (compresi i valori degli eventi futuri).

Per i filtri *country_code* e *currency*, i valori NULL e "" sono equivalenti e significano l'assenza del filtro.

Per *country_code*, deve essere utilizzato il campo *code* della struttura MqlCalendarCountry, ad esempio "US", "RU" o "EU".

Per *currency*, deve essere utilizzato il campo *currency* della struttura MqlCalendarCountry, ad esempio "USD", "RUB" o "EUR".

I filtri sono applicati per congiunzione, es. l' 'AND' logico viene utilizzato per selezionare solo i valori degli eventi per cui vengono soddisfatte contemporaneamente entrambe le condizioni (Paese e valuta).

Se un valore di un evento non ha nessuno dei campi specificati di seguito

```
struct MqlCalendarValue
  {
   ...
   long              actual_value;              // valore attuale dell'evento
   long              prev_value;                // valore precedente dell'evento
   long              revised_prev_value;        // valore rivisto dell'evento pr
   long              forecast_value;            // valore di previsione dell'eve
   ...
  };
```

allora il valore del campo mancante viene restituito come INT64_MIN (-9223372036854775808). Vedere il valore campo *revised_prev_value* nell'esempio seguente.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- codice Paese per EU (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- ottiene tutti i valori degli eventi EU
   MqlCalendarValue values[];
//--- imposta i limiti dell'intervallo da cui prendiamo gli eventi
   datetime date_from=D'01.01.2018';  // prende tutti gli eventi dal 2018
   datetime date_to=0;                // 0 indica tutti gli eventi noti, inclusi quel
//--- richiede la cronologia degli eventi dell'UE dall'anno 2018
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
    {
      PrintFormat("Valori evento ricevuti per country_code=%s: %d",
```

```
                  EU_code,ArraySize(values));
      //--- riduce la grandezza dell'array per l'output sul Journal
      ArrayResize(values,10);
//--- visualizza i valori degli eventi nel Journal
      ArrayPrint(values);
     }
   else
    {
      PrintFormat("Errore! Impossibile ricevere eventi per country_code=%s",EU_code);
      PrintFormat("Codice errore: %d",GetLastError());
     }
//---
  }
/*
  Risultato:
  Valori evento ricevuti per country_code=EU: 1384
        [id] [event_id]           [time]               [period] [revision]    [actual_v
   [0] 54215  999500001 2018.01.02 09:00:00 2017.12.01 00:00:00           3        60600
   [1] 54221  999500002 2018.01.04 09:00:00 2017.12.01 00:00:00           3        56600
   [2] 54222  999500003 2018.01.04 09:00:00 2017.12.01 00:00:00           3        58100
   [3] 45123  999030005 2018.01.05 10:00:00 2017.11.01 00:00:00           0          600
   [4] 45124  999030006 2018.01.05 10:00:00 2017.11.01 00:00:00           0         2800
   [5] 45125  999030012 2018.01.05 10:00:00 2017.12.01 00:00:00           1          900
   [6] 45126  999030013 2018.01.05 10:00:00 2017.12.01 00:00:00           1         1400
   [7] 54953  999520001 2018.01.05 20:30:00 2018.01.02 00:00:00           0       127900
   [8] 22230  999040003 2018.01.08 10:00:00 2017.12.01 00:00:00           0         9100
   [9] 22231  999040004 2018.01.08 10:00:00 2017.12.01 00:00:00           0        18400
*/
```

Guarda anche

CalendarCountries, CalendarEventByCountry, CalendarValueHistoryByEvent, CalendarEventById, CalendarValueById

CalendarValueLastByEvent

Ottiene l'array di valori di eventi in base al suo ID poiché lo stato del database di Calendar con un change_id specificato.

```
int  CalendarValueLastByEvent(
   ulong                event_id,      // ID evento
   ulong&               change_id,     // valore ID evento
   MqlCalendarValue&    values[]       // array per le descrizioni dei valori
   );
```

Parametri

event_id

[in] Event ID.

change_id

[in][out] Change ID.

values[]

[out] MqlCalendarValue: è il tipo di array per ricevere i valori degli eventi

Valore di ritorno

Numero di valori di eventi ricevuti. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insufficiente per l'esecuzione di una richiesta),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto),
- 5400 - ERR_CALENDAR_MORE_DATA (la dimensione dell'array è insufficiente per la ricezione delle descrizioni di tutti i valori, solo quelli che sono riusciti a rientrare sono stati ricevuti),
- errori di esecuzione fallita di ArrayResize()

Nota

Tutte le funzioni per lavorare con il calendario economico utilizzano l'ora del trade server (TimeTradeServer). Ciò significa che il tempo nella struttura MqlCalendarValue e gli input di tempo nelle funzioni CalendarValueHistoryByEvent/CalendarValueHistory sono impostate nel fuso orario del trade server, piuttosto che nell'ora locale dell' utente.

Se l'array *events[]* di lunghezza fissa è stato passato alla funzione e non c'era spazio sufficiente per salvare l'intero risultato, l'errore ERR_CALENDAR_MORE_DATA (5400) viene attivato.

Se *change_id* = 0 viene passato alla funzione, la funzione restituisce sempre zero ma il database del calendario corrente viene restituito a *change_id*.

La funzione restituisce l'array per una notizia specificata ed un nuovo *change_id* che può essere utilizzato per le chiamate successive della funzione per ricevere i nuovi valori della notizia. Pertanto, è possibile aggiornare i valori per una notizia specificata chiamando questa funzione con l'ultimo *change_id* noto.

Se un valore di un evento non ha nessuno dei campi specificati di seguito

```
    struct MqlCalendarValue
```

```
      {
       ...
       long              actual_value;              // valore attuale dell'evento
       long              prev_value;                // valore precedente dell'evento
       long              revised_prev_value;        // valore rivisto dell'evento pr
       long              forecast_value;            // valore di previsione dell'eve
       ...
      };
```

quindi il valore del campo mancante viene restituito come INT64_MIN (-9223372036854775808).

L' EA di esempio che ascolta il rilascio del report Nonfarm payrolls:

```
#property description "Esempio di utilizzo della funzione CalendarValueLastByEvent"
#property description " per tenere traccia del rilascio del report Nonfarm Payrolls"
#property description "Per ottenere ciò, ottieni l'ID della modifica corrente"
#property description " del database del Calendario. Quindi, usa questo ID per ricever
#property description " solo nuovi eventi tramite il sondaggio del timer"
//+------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                        |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crea il timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di deinitializzazione dell' Expert                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- distruggi il timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Funzione tick dell' Expert                                       |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Funzione del timer                                               |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- Cambia ID del database del calendario
```

```
   static ulong calendar_change_id=0;
//--- primo attributo di lancio
   static bool first=true;
//--- ID evento
   static ulong event_id=0;
//--- nome dell'evento
   static string event_name=NULL;
//--- arra del valore di evento
   MqlCalendarValue values[];
//--- esegue l'inizializzazione - ottiene l'attuale calendar_change_id
   if(first)
    {
      MqlCalendarEvent events[];
      //--- codice Paese per gli USA (ISO 3166-1 Alpha-2)
      string USA_code="US";
      //--- ottieni eventi per gli USA
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- posizione di un evento necessario nell' array 'events'
      int event_pos=-1;
      //--- visualizza gli eventi USA nel Journal
      if(events_count>0)
        {
         PrintFormat("%s: eventi USA: %d",__FUNCTION__,events_count);
         for(int i=0;i<events_count;i++)
          {
            string event_name_low=events[i].name;
            //--- cambia il nome di un evento in minuscolo
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() ha restituito l'errore %d",GetLastError())
               //--- esce dalla funzione prima del tempo
               return;
              }
            //--- cerca l'evento "Nonfarm Payrolls"
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- evento trovato, ricorda il suo ID
               event_id=events[i].id;
               //--- scrive il nome dell'evento "Nonfarm Payrolls"
               event_name=events[i].name;
               //--- ricorda la posizione degli eventi nell' array 'events[]'
               event_pos=i;
               //--- tieni presente che il Calendario presenta diversi eventi contener
               PrintFormat("Evento \"Nonfarm Payrolls\" trovato: event_id=%d  event_na
               //--- visualizza tutti gli eventi commentando l'operatore 'break' per
               break;
              }
          }
        //--- riduci la lista cancellando gli eventi dopo "Nonfarm Payrolls"
```

```
         ArrayRemove(events,event_pos+1);
         //--- lascia 9 eventi prima di "Nonfarm Payrolls" per analisi più convenienti
         ArrayRemove(events,0,event_pos-9);
         ArrayPrint(events);
        }
      else
        {
         PrintFormat("%s: CalendarEventByCountry(%s) ha restituito 0 eventi, codice er
                     USA_code,__FUNCTION__,GetLastError());
         //--- operazione completata in errore, riprovare durante la prossima chiamata
         return;
        }

      //--- ottiene l'ID di modifica del database di Calendar per l'evento specificato
      if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
        {
         //--- questo blocco di codice non può essere eseguito durante il primo avvio,
         PrintFormat("%s: Ricevuto l'ID corrente del database del calendario: change_i
                     __FUNCTION__,calendar_change_id);
         //--- imposta la flag ed esce prima del prossimo evento del timer
         first=false;
         return;
        }
      else
        {
         //--- i dati non vengono ricevuti (questo è normale per il primo avvio), cont
         int error_code=GetLastError();
         if(error_code==0)
          {
           PrintFormat("%s: Ricevuto l'ID corrente del database del calendario: chang
                       __FUNCTION__,calendar_change_id);
           //--- imposta la flag ed esce prima del prossimo evento del timer
           first=false;
           //--- ora abbiamo il valore calendar_change_id
           return;
          }
         else
          {
           //--- e questo è davvero un errore
           PrintFormat("%s: Fallimento nell'ottenere valori per event_id=%d",__FUNCTI
           PrintFormat("Codice errore: %d",error_code);
           //--- operazione completata con errore, riprovare durante la prossima chia
           return;
          }
        }
     }

// --- abbiamo l'ultimo valore conosciuto del change ID del calendario (change_id)
   ulong old_change_id=calendar_change_id;
```

```
//--- controlla un nuovo valore dell' evento Nonfarm Payrolls
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
    {
      PrintFormat("%s: Ricevuti nuovi eventi per \"%s\": %d",
                  __FUNCTION__,event_name,ArraySize(values));
      //--- visualizza i dati dall' arra 'values' nel Journal
      ArrayPrint(values);
      //--- mostra i valori del precedente e del nuovo ID del diario
      PrintFormat("%s: Precedente change_id=%d, nuovo change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      scrivi il tuo codice che deve gestire la data di rilascio di "Nonfarm Payrolls"c
      */
     }
//---
  }
/*
  Risultato:
   OnTimer: USA events: 202
  Evento "Nonfarm Payrolls" trovato: event_id=840030016  event_name=Nonfarm Payrolls
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importar
   [0] 840030007      1        4           2           0           840      1
   [1] 840030008      1        4           2           0           840      1
   [2] 840030009      1        4           2           0           840      0
   [3] 840030010      1        4           2           0           840      0
   [4] 840030011      1        4           2           0           840      1
   [5] 840030012      1        4           2           0           840      1
   [6] 840030013      1        4           2           0           840      1
   [7] 840030014      1        4           2           0           840      1
   [8] 840030015      1        3           2           0           840      1
   [9] 840030016      1        3           2           0           840      4
   OnTimer: Ha ricevuto l'ID corrente del database del Calendar: change_id=33986560

*/
```

Guarda anche

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

CalendarValueLast

Ottiene l'array di valori per tutti gli eventi con la possibilità di ordinare per Paese e/o valuta dallo stato del database del calendario con un change_id specificato.

```
int  CalendarValueLast(
   ulong&               change_id,             // change ID
   MqlCalendarValue&    values[],              // array per le descrizioni dei valori
   const string         country_code=NULL,     // nome codice Paese (ISO 3166-1 alpha-
   const string         currency=NULL          // nome del codice valuta del Paese
   );
```

Parametri

change_id

[in][out] Change ID.

values[]

[out] MqlCalendarValue: è il tipo di array per ricevere i valori degli eventi

country_code=NULL

[in] Nome codice Paese (ISO 3166-1 alpha-2)

currency=NULL

[in] Nome del codice valuta del Paese.

Valore di ritorno

Numero di valori di eventi ricevuti. Per ottenere informazioni su un errore, chiamare la funzione GetLastError(). Possibili errori:

- 4001 - ERR_INTERNAL_ERROR (errore di runtime generale),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memoria insufficiente per l'esecuzione di una richiesta),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite della richiesta ecceduto),
- 5400 - ERR_CALENDAR_MORE_DATA (la dimensione dell'array è insufficiente per la ricezione delle descrizioni di tutti i valori, solo quelli che sono riusciti a rientrare sono stati ricevuti),
- errori di esecuzione fallita di ArrayResize()

Nota

Tutte le funzioni per lavorare con il calendario economico utilizzano l'ora del trade server (TimeTradeServer). Ciò significa che il tempo nella struttura MqlCalendarValue e gli input di tempo nelle funzioni CalendarValueHistoryByEvent/CalendarValueHistory sono impostate nel fuso orario del trade server, piuttosto che nell'ora locale dell' utente.

Se l'array *events[]* di lunghezza fissa è stato passato alla funzione e non c'era spazio sufficiente per salvare l'intero risultato, l'errore ERR_CALENDAR_MORE_DATA (5400) viene attivato.

Se *change_id* = 0 viene passato alla funzione, si otterrà il corrente *change_id* del database del calendario su quel parametro; e la funzione restituisce 0

Per i filtri *country_code* e *currency*, i valori NULL e "" sono equivalenti e significano l'assenza del filtro.

Per *country_code*, deve essere utilizzato il campo *code* della struttura MqlCalendarCountry, ad esempio "US", "RU" o "EU".

Per *currency*, deve essere utilizzato il campo *currency* della struttura MqlCalendarCountry, ad esempio "USD", "RUB" o "EUR".

I filtri sono applicati per congiunzione, es. l' 'AND' logico viene utilizzato per selezionare solo i valori degli eventi per cui vengono soddisfatte contemporaneamente entrambe le condizioni (paese e valuta)

La funzione restituisce l'array per una notizia specificata ed un nuovo *change_id* che può essere utilizzato per le chiamate successive della funzione per ricevere i nuovi valori della notizia. Pertanto, è possibile aggiornare i valori per una notizia specificata chiamando questa funzione con l'ultimo *change_id* noto.

Se un valore di un evento non ha nessuno dei campi specificati di seguito

```
struct MqlCalendarValue
  {
   ...
   long              actual_value;              // valore attuale dell'evento
   long              prev_value;                // valore precedente dell'evento
   long              revised_prev_value;        // valore rivisto dell'evento pr
   long              forecast_value;            // valore di previsione dell'eve
   ...
  };
```

quindi il valore del campo mancante viene restituito come INT64_MIN (-9223372036854775808).

L' EA di esempio che ascolta gli eventi del calendario economico:

```
#property description "Esempio di utilizzo della funzione CalendarValueLast"
#property description " per sviluppare l'ascoltatore di eventi del calendario economic
#property description "Per ottenere ciò, ottieni l'ID della modifica corrente"
#property description " del database del Calendario. Quindi, usa questo ID per riceve
#property description " solo nuovi eventi tramite il sondaggio del timer"
//+------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                        |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- crea il timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di deinitializzazione dell' Expert                      |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- distruggi il timer
```

```
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Funzione tick dell' Expert                                       |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Funzione del timer                                               |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- Cambia ID del database del calendario
   static ulong calendar_change_id=0;
//--- primo attributo di lancio
   static bool first=true;
//--- arra del valore di evento
   MqlCalendarValue values[];
//--- esegue l'inizializzazione - ottiene l'attuale calendar_change_id
   if(first)
    {
// --- ottiene l'ID di modifica del database del Calendario
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- questo blocco di codice non può essere eseguito durante il primo avvio,
         PrintFormat("%s: Ricevuto l'ID corrente del database del calendario: change_i
                     __FUNCTION__,calendar_change_id);
         //--- imposta la flag ed esce prima del prossimo evento del timer
         first=false;
         return;
        }
      else
        {
         //--- i dati non vengono ricevuti (questo è normale per il primo avvio), cont
         int error_code=GetLastError();
         if(error_code==0)
          {
           PrintFormat("%s: Ricevuto l'ID corrente del database del calendario: chang
                       __FUNCTION__,calendar_change_id);
           //--- imposta la flag ed esce prima del prossimo evento del timer
           first=false;
           //--- ora abbiamo il valore calendar_change_id
           return;
          }
         else
          {
```

```
            //--- e questo è davvero un errore
            PrintFormat("%s: impossibile ottenere eventi in CalendarValueLast. Codice
                        __FUNCTION__,error_code);
            //--- operazione completata in caso di errore, reinizializzazione durante
            return;
           }
        }
     }

// --- abbiamo l'ultimo valore conosciuto del change ID del calendario (change_id)
   ulong old_change_id=calendar_change_id;
//--- controlla se ci sono nuovi eventi del calendario
   if(CalendarValueLast(calendar_change_id,values)>0)
    {
      PrintFormat("%s: nuovi eventi del calendario ricevuti: %d",
                  __FUNCTION__,ArraySize(values));
      //--- visualizza i dati dall' arra 'values' nel Journal
      ArrayPrint(values);
      //--- mostra i valori del precedente e del nuovo ID del diario
      PrintFormat("%s: Precedente change_id=%d, nuovo change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- mostra nuovi eventi nel Journal
      ArrayPrint(values);
      /*
     scrivi il tuo codice per gestire il verificarsi di eventi qui
      */
     }
//---
  }
/*
  Esempio dell'operazione listener(di ascolto):
  OnTimer: ha ricevuto l'ID corrente del database del calendario: change_id=33281792
  On Timer: Ricevuti nuovi eventi per il calendario: 1
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  OnTimer: Previous change_id=33281792, new change_id=33282048
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  On Timer: Ricevuti nuovi eventi per il calendario: 1
        [id] [event_id]              [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203
  OnTimer: Previous change_id=33282048, new change_id=33282560
        [id] [event_id]              [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203

*/
```

Guarda anche

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

Accesso alle Timeseries e ad i dati degli Indicatori

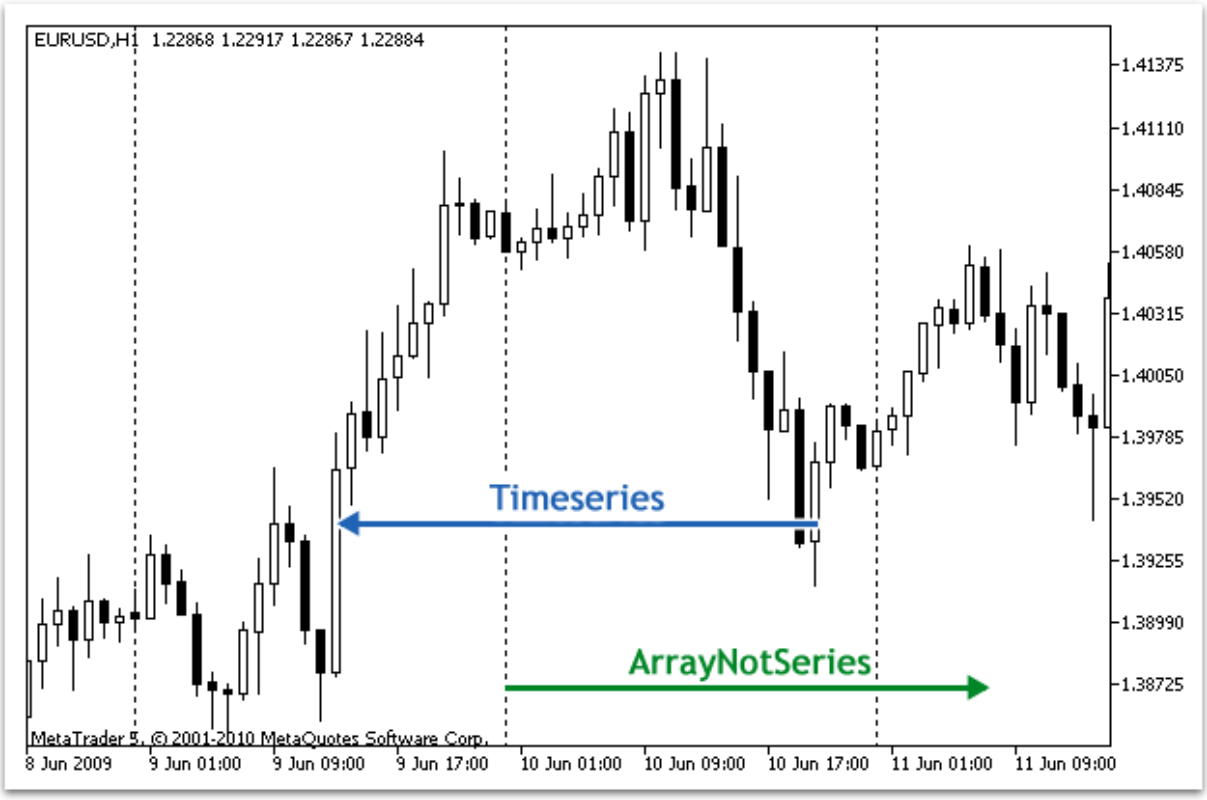
Si tratta di funzioni per lavorare con le serie temporali (_* Timeseries) e con gli indicatori. Una timeseries si differenzia dal consueto array di dati, per via del suo ordinamento inverso - elementi delle timeseries sono indicizzati a partire dalla fine di un array fino al suo inizio (a partire dai dati più recenti a quelli più vecchi). Per copiare i valori delle time-series degli indicatori e dei dati, si consiglia di utilizzare solo array dinamici, perché le funzioni di copiatura sono progettate per assegnare la grandezza necessaria degli array che ricevono i valori.

C'è un' **importante eccezione** a questa regola: se i valori delle timeseries e degli indicatori devono essere copiati spesso, ad esempio ad ogni chiamata di OnTick() negli Expert Advisors o ad ogni chiamata di OnCalculate() negli indicatori, in questo caso sarebbe meglio utilizzare gli array staticamente distribuiti, perché **operazioni di allocazione di memoria** per gli array dinamici **richiedono più tempo**, e ciò avrà un effetto durante il testing e l'ottimizzazione.

Quando si utilizzano funzioni che accedono ai valori di timeseries e indicatori, la direzione di indicizzazione dovrebbe essere presa in considerazione. Questo è descritto nella sezione Direzione di indicizzazione negli e nelle timeseries.

L'accesso ai dati dell' indicatore e timeseries è implementato indipendentemente dal fatto se i dati richiesti sono pronti (il cosiddetto accesso asincrono). Questo è estremamente importante per il calcolo dell' indicatore personalizzato, quindi se non vi sono dati, funzioni di tipo *Copy...()* restituiscono immediatamente un errore. Tuttavia, quando si accede da Expert Advisor e script, diversi tentativi di ricezione dei dati vengono realizzati in una piccola pausa, che è volta a fornire un po' di tempo necessario per scaricare TimeSeries richieste per calcolare i valori degli indicatori.

La sezione Organizzare l' Accesso ai Dati descrive i dettagli di ricezione, archiviazione e richiede i dati sui prezzi nel terminale client MetaTrader 5.



È storicamente accettato che un accesso ai dati dei prezzi in un array viene eseguito a partire dalla fine dei dati. Fisicamente, i nuovi dati vengono sempre scritti alla fine dell'array, ma l'indice della matrice è sempre uguale a zero. L'indice 0 nell'array timeseries denota i dati della barra corrente, cioè la barra che corrisponde all'intervallo di tempo non-finito in questo timeframe.

Un timeframe è il periodo di tempo, durante il quale viene formata una barra di prezzo unico. Ci sono 21 timeframes standard predefiniti.

Funzione	Azione
SeriesInfoInteger	Restituisce le informazioni sullo stato dei dati storici
Bars	Restituisce il numero di barre dello storico di un simbolo e periodo specificati
BarsCalculated	Restituisce il numero di dati calcolati in un buffer indicatore, oppure -1 in caso di errore (dati non sono stati ancora calcolati)
IndicatorCreate	Restituisce l'handle per l'indicatore tecnico specificato creato da una serie di parametri di tipo MqlParam
IndicatorParameters	Basato sull' handler specificato, restituisce il numero di parametri di input dell'indicatore, nonché i valori e tipi dei parametri
IndicatorRelease	Rimuove un handle indicatore e rilascia il blocco di calcolo dell'indicatore, se non è usato da nessun altro
CopyBuffer	Mette in un array i dati di un buffer specificato da un indicatore specificato
CopyRates	Mette in un array i dati storici della struttura Rates per un simbolo e periodo specificati
CopySeries	Ottiene le Timeseries sincronizzate dalla struttura MqlRates per il simbolo-periodo specificato e la quantità specificata. I dati vengono ricevuti nella serie di array indicati
CopyTime	Mette in un array i dati storici dell'orario di apertura della barra per un simbolo e periodo specificati
CopyOpen	Mette in un array i dati storici sul prezzo di apertura della barra per un simbolo e periodo specificati
CopyHigh	Mette in un array i dati storici sul prezzo massimo della barra per un periodo e simbolo specificati
CopyLow	Mette in un array i dati storici sul prezzo minimo della barra per un simbolo e periodo specificati
CopyClose	Mette in un array i dati storici sul prezzo di chiusura della barra per un simbolo e periodo specificati
CopyTickVolume	Mette in un array i dati storici relativi ai volumi tick per un simbolo e periodo specificati
CopyRealVolume	Mette in un array i dati storici sui volumi trade per un simbolo e periodo specificati

Funzione	Azione
CopySpread	Mette in un array i dati storici sugli spread per un simbolo e periodo specificati
CopyTicks	La funzione riceve ticks nel formato MqlTick in ticks_array
CopyTicksRange	La funzione riceve ticks in formato MqlTick nell'intervallo di date specificato in ticks_array
iBars	Restituisce il numero di barre di un simbolo e di un periodo corrispondenti, disponibili nello storico
iBarShift	Restituisce l'indice della barra corrispondente al tempo/orario specificato
iClose	Restituisce il prezzo Close (di chiusura) della barra (indicato dal parametro 'shift') sul chart corrispondente
iHigh	Restituisce il prezzo High della barra (indicata dal parametro 'shift') sul chart corrispondente
iHighest	Restituisce l'indice del valore più alto trovato sul chart corrispondente (spostamento relativo alla barra corrente)
iLow	Restituisce il prezzo Low della barra (indicato dal parametro 'shift') sul chart corrispondente
iLowest	Restituisce l'indice del valore più piccolo trovato sul chart corrispondente (slittamento relativo alla barra corrente)
iOpen	Restituisce il prezzo Open della barra (indicata dal parametro 'shift') sul chart corrispondente
iTime	Restituisce il tempo di apertura della barra (indicato dal parametro 'shift') sul chart corrispondente
iTickVolume	Restituisce il volume tick della barra (indicato dal parametro 'shift') sul chart corrispondente
iRealVolume	Restituisce il volume reale della barra (indicato dal parametro 'shift') sul chart corrispondente
iVolume	Restituisce il volume tick della barra (indicato dal parametro 'shift') sul chart corrispondente
iSpread	Restituisce il valore di spread della barra (indicato dal parametro 'shift') sul chart corrispondente

Nonostante il fatto che con la funzione ArraySetAsSeries() è possibile impostare negli array l'accesso agli elementi come quelli nelle timeseries, va ricordato che gli elementi dell'array vengono fisicamente memorizzati in un solo e medesimo ordine - solo la direzione di indicizzazione cambia. Per dimostrare questo fatto cerchiamo di fare un esempio:

```
   datetime TimeAsSeries[];
//--- imposta l' accesso all'array come ad una timeseries
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
```

```
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
   if(copied<=0)
     {
      Print("L'operazione di copia dei valori dell'orario di apertura per le ultime 1(
      return;
     }
   Print("TimeCurrent =",TimeCurrent());
   Print("ArraySize(Time) =",ArraySize(TimeAsSeries));
   int size=ArraySize(TimeAsSeries);
   for(int i=0;i<size;i++)
     {
      Print("TimeAsSeries["+i+"] =",TimeAsSeries[i]);
     }

   datetime ArrayNotSeries[];
   ArraySetAsSeries(ArrayNotSeries,false);
   ResetLastError();
   copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
   if(copied<=0)
     {
      Print("L'operazione di copia dei valori dell'orario di apertura per le ultime 1(
      return;
     }
   size=ArraySize(ArrayNotSeries);
   for(int i=size-1;i>=0;i--)
     {
      Print("ArrayNotSeries["+i+"] =",ArrayNotSeries[i]);
     }
```

Come risultato si otterrà un output come questo:

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
```

```
ArrayNotSeries[4] = 2009.06.11 09:00:00
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

Come si vede dall' output, così come aumenta l'indice dell'array TimeAsSeries, il valore temporale dell'indice diminuisce, cioè si passa dal presente al passato. Per l'array comune ArrayNotSeries il risultato è diverso - cos come indice cresce, si passa dal passato al presente.

Vedere anche

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

Direzione di Indicizzazione negli Array, Buffers e TimeSeries

L'indicizzazione predefinita di tutti gli array e buffer indicatore è da sinistra a destra. The index of the first element is always equal to zero. Così, il primo elemento di un buffer di array o indicatore con indice 0 è predefinito sulla posizione estrema sinistra, mentre l'ultimo elemento è sulla posizione estrema destra.

Un buffer indicatore è un array dinamico di tipo doppio, la cui grandezza è gestita dai terminali client, in modo che corrisponda sempre al numero di barre su cui viene calcolato l'indicatore. Un solito array dinamico di tipo double viene assegnato come un buffer utilizzando l'indicatore funzione SetIndexBuffer(). I buffers indicatore non richiedono l'impostazione delle loro grandezze con la funzione ArrayResize() - questo sarà fatto dal sistema di esecuzione del terminale.

Timeseries sono array con indicizzazione inversa, cioè il primo elemento di un timeseries è nella posizione estrema destra, e l'ultimo elemento è nella posizione di estrema sinistra. Timeseries vengno utilizzate per la memorizzazione di dati sui prezzi storici e contenere le informazioni di tempo, possiamo dire che i dati più recenti vengono inseriti nella posizione estrema destra delle timeseries, mentre i dati più vecchi sono in posizione di estrema sinistra.

Quindi l'elemento timeseries con indice 0 contiene le informazioni sulle ultime quotazioni di un simbolo. Se una timeseries contiene dati su un timeframe giornaliero, i dati del giorno corrente ancora incompleto si trovano sulla posizione zero, e la posizione di indice 1 contiene dati di ieri.

Modifica della Direzione di Indicizzazione

La Funzione ArraySetAsSeries() permette di cambiare il metodo di accesso agli elementi di un array dinamico, l'ordine fisico di memorizzazione dei dati nella memoria del computer non viene cambiato da questo. Questa funzione cambia semplicemente il metodo di indirizzazione degli elementi dell'array, in modo che quando si copia un array a un'altro utilizzando la funzione ArrayCopy(), il contenuto dell' array destinatario non dipende dalla direzione di indicizzazione nell' array di origine.

La direzione di indicizzazione non può essere modificata per array staticamente distribuiti. Anche se un array viene passato come parametro ad una funzione, il tentativo di cambiare la direzione di indicizzazione all'interno di questa funzione non porterà alcun effetto.

Per i buffer indicatore, come per gli array solitamente, la direzione di indicizzazione può anche essere impostato come invertita (come nelle timeseries), cioè il riferimento alla posizione zero nel buffer indicatore significherà riferimento all'ultimo valore sul buffer indicatore corrispondente e questo corrisponderà al valore dell'indicatore sull'ultima barra. Tuttavia, la posizione fisica delle barre indicatori non verrà modificata.

Ricezione di Dati Prezzi negli Indicatori

Ogni indicatore personalizzato deve necessariamente contenere la funzione OnCalculate(), a cui dati sui prezzi richiesti per il calcolo dei valori di buffer indicatori vengono passati. La direzione di indicizzazione in questi array passati può essere individuata con la funzione ArrayGetAsSeries().

Gli Array passati alla funzione riflettono dati sui prezzi, vale a dire questi array hanno il segno di una timeseries e la funzione ArrayIsSeries() restituirà true quando si controllano questi array. Tuttavia, qualsiasi direzione indicizzazione dovrebbe essere controllata solo per la funzione ArrayGetAsSeries().

Per non dipendere da valori predefiniti, ArraySetAsSeries() deve essere incondizionatamente chiamato per gli array che si stanno per lavorare, ed impostare la direzione desiderata.

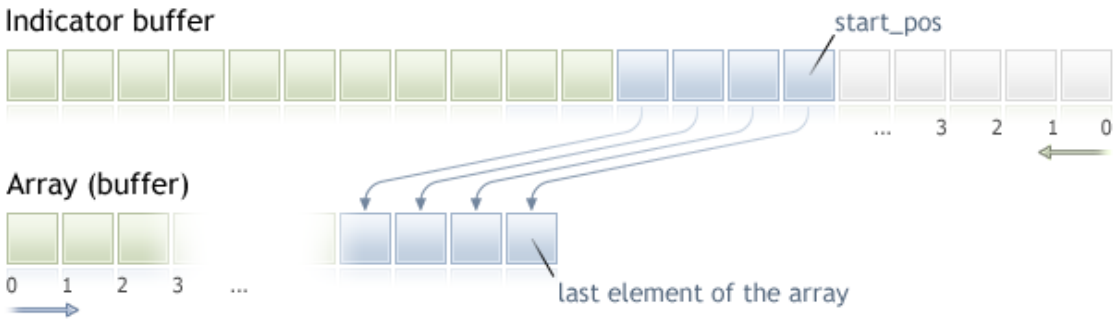
Ricezione di Dati Prezzi e Valori degli Indicatori

La direzione di indicizzazione di default di tutti gli array negli Expert Advisor, indicatori e script è da sinistra a destra. Se necessario, in qualsiasi programma mql5 è possibile richiedere valori timeseries su qualsiasi simbolo e timeframe, così come i valori degli indicatori calcolati su qualsiasi simbolo e timeframe.

Utilizzare le funzioni Copy...() per questi scopi:

- CopyBuffer - copia i valori di un buffer indicatore in un array di tipo double;
- CopyRates - copia lo storico prezzi in un array di strutture MqlRates;
- CopyTime - copia i valori Time in un array di tipo datetime;
- CopyOpen - copia i valori Open in un array di tipo double;
- CopyHigh - copia i valori High in un array di tipo double;
- CopyLow - copia i valori Low in un array di tipo double;
- CopyClose - copia i valori Close in un array di tipo double;
- CopyTickVolume - copia i volumi tick volumi in un array di tipo long;
- CopyRealVolume - copia i volumi equity in un array di tipo long;
- CopySpread - copia la cronistoria dello spread in un array di tipo int;

Tutte queste funzioni funzionano in modo simile. Prendiamo in considerazione i dati ottenendo meccanismo sull'esempio di CopyBuffer(). È implicato che la direzione di indicizzazione dei dati richiesti è quella delle timeseries, e la posizione di indice 0 (zero) memorizza i dati della barra corrente ancora incompleta. Al fine di ottenere l'accesso a questi dati abbiamo bisogno di copiare il volume necessario di dati nell' array ricevente, ad esempio in un array *buffer*.



Quando si copia è necessario specificare la posizione di partenza nell' array di origine, a partire dal quale i dati saranno copiati nell' array destinatario. In caso di successo, il numero specificato di elementi verrà copiato nell' array destinatario dall' array di origine (dal buffer indicatore in questo caso). Indipendentemente dal valore di indicizzazione nell'array destinatario, la copia viene sempre eseguita come è mostrato nella figura precedente.

Se si prevede che i dati sui prezzi saranno trattati in un ciclo con un gran numero di iterazioni, si consiglia di controllare il fatto di cessazione forzata del programma usando la funzione IsStopped():

```
int copied=CopyBuffer(ma_handle,// Handle indicatore
                      0,        // L'indice del buffer dell'indicatore
                      0,        // Posizione iniziale per la copia
                      number,   // Numero di valori da copiare
                      Buffer    // L'array che riceve i valori
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- Prendi il valore per l'indice k
   double value=Buffer[k];
   // ...
   // lavora con il valore
   k++;
  }
```

Esempio:

```
input int per=10; // periodo dell'esponente
int ma_handle;    // Handle indicatore
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                    |
//+------------------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// Handle indicatore
                         0,        // indice del buffer indicatore
                         0,        // posizione di partenza da cui copiare
                         10,       // numero di valori per la copia
                         10,       // array ricevente valori
                         );
   if(copied<0) return;
// .... ulteriore codice
  }
```

Vedi anche

Organizzazione di Accesso ai Dati

Organizzazione di Accesso ai Dati

In this section questions connected with obtaining, storing and requesting price data (timeseries) are considered.

Ricezione dei Dati da un Trade Server

Prima che i dati sui prezzi saranno disponibili nel terminale MetaTrader 5, devono essere ricevuti ed elaborati. Per ricevere i dati, deve essere stabilita la connessione al trade server MetaTrader 5. I dati vengono ricevuti sotto forma di blocchi compatti di barre minute, dal server su richiesta del terminale.

Il meccanismo di riferimento server per i dati non dipende da come la richiesta è stata avviata - da un utente durante la navigazione nel grafico o in modo con il programma nella linguaggio MQL5.

Memorizzazione di Dati Intermedi

I dati ricevuti da un server vengono automaticamente decompressi e salvati nella formato intermedio HCC. I dati su ogni simbolo vengono scritti in una cartella separata: *terminal_directory*\bases*server_name*\history*symbol_name*. Per esempio, i dati relativi ad EURUSD ricevuti dal server MetaQuotes-Demo verranno memorizzati in *terminal_directory*\bases\MetaQuotes-Demo\history\EURUSD\.

I dati vengono scritti in file con estensione .hcc . Ogni file memorizza i dati delle barre minute per un anno. Ad esempio, il file denominato 2009.hcc nella cartella EURUSD contiene barre minute di EURUSD per l'anno 2009. Questi file vengono utilizzati per la preparazione di dati sui prezzi per tutti i timeframes e non sono destinati per l'accesso diretto.

Recupero dei Dati in un Timeframe Necessario su Dati Intermedi

File intermedi HCC vengono utilizzati come fonte di dati per la creazione di dati sui prezzi per i timeframe richiesti nel formato HCC. I dati di formato HCC sono timeseries che sono preparate massimalmente per un accesso rapido. Essi vengono creati su richiesta di un grafico o di un programma MQL5. Il volume dei dati non deve superare il valore del parametro "Max barre nei grafici". I dati sono conservati per un ulteriore utilizzo in file con estensione hcc.

Per risparmiare le risorse, i dati su un timeframe vengono memorizzati e salvati nella RAM solo se necessario. Se non vengono chiamati per lungo tempo, vengono rilasciati dalla RAM e salvati in un file. Per ogni timeframe, i dati vengono preparati indipendentemente dal fatto che ci sono dati pronti per altri timeframes o meno. Le regole di formazione ed accesso ai dati sono gli stessi per tutti i timeframes. Significa, che nonostante il fatto che i dati memorizzati in unità HCC siano nn minuto, la disponibilità di dati HCC non significa la disponibilità di dati sulla tempistica M1 come HC nello stesso volume.

La ricezione di nuovi dati da un server chiama l'aggiornamento automatico dei dati sui prezzi utilizzati in formato HC di tutti i timeframes. Essa comporta anche il ricalcolo di tutti gli indicatori che implicitamente li utilizzano come dati di input per i calcoli.

Parametro "Max barre nel grafico"

Il parametro "Max barre nei grafici" limita il numero di barre in formato HC a disposizione di grafici, indicatori e programmi MQL5. Questo è valido per tutti i timeframe disponibili e serve, prima di tutto, per risparmiare risorse informatiche.

Quando si imposta un valore elevato di questo parametro, va ricordato, che se sono disponibili dati storici in profondità per piccoli timeframes, la memoria utilizzata per la memorizzazione dei buffers di timeseries ed indicatori, possono diventare centinaia di megabyte e raggiungere la restrizione RAM per il programma del terminale client (2Gb per applicazioni a 32 bit di MS Windows).

La modifica di "Max barre nei grafici" avrà effetto dopo che il terminale client viene riavviato. Il cambio di questo parametro causa né il riferimento automatico ad un server per i dati aggiuntivi, né formazione di barre aggiuntive di di timeseries. Dati relativi ai prezzi aggiuntivi vengono richiesti dal server, e le timeseries vengono aggiornate tenendo conto della nuova limitazione, in caso di uno scorrimento del grafico all'area senza dati, o quando i dati vengono richiesti da un programma MQL5.

Volume dei dati richiesti dal server corrisponde al numero di barre richiesto di questo timeframe con il parametro "Max barre nel grafico" preso in considerazione. La restrizione impostata da questo parametro non è rigida, e in alcuni casi il numero di barre disponibili per un periodo di tempo può essere un po' di più del valore corrente del parametro.

Disponibilità dati

La presenza di dati su formato HCC o anche in preparazione per l'utilizzo del formato HC, non sempre denota la disponibilità assoluta di tali dati da visualizzare nel chart o da utilizzare in programmi MQL5.

Quando si accede ai dati sui prezzi o valori degli indicatori da un programma MQL5 va ricordato che la loro disponibilità in un certo istante di tempo o a partire da un certo momento di tempo non è garantita. E' collegato con il fatto che con il fine di risparmiare risorse, la copia completa di dati necessari per un programma MQL5 non è memorizzata in MetaTrader 5; viene dato solo l'accesso diretto al database terminale.

La cronistoria dei prezzi per tutti i timeframes è costruita a partire da dati comuni di formato HCC, e qualsiasi aggiornamento dei dati da un server conduce all'aggiornamento dei dati per tutti i timeframes e al ricalcolo degli indicatori. A causa di tale accesso ai dati, può essere chiusa, anche se questi dati erano disponibili poc'anzi.

Sincronizzazione dei Dati Terminale e Dati Server

Sicchè un programma MQL5 può richiamare dati da qualsiasi simbolo e timeframe, vi è la possibilità che i dati di una timeseries necessari non sono ancora formati nel terminale o i dati sui prezzi necessari non sono sincronizzati con il trade server. In questo caso è difficile prevedere il tempo di latenza.

Gli algoritmi con cicli di latenza non sono la soluzione migliore. L'unica eccezione in questo caso sono gli script, perché non hanno nessun algoritmo alternativo a causa di non avere la gestione degli eventi (event handling). Per gli indicatori personalizzati, tali algoritmi, nonché eventuali altri cicli di latenza sono fortemente sconsigliati, perché portano alla cessazione del calcolo di tutti gli indicatori e qualsiasi altra manipolazione dei dati sui prezzi del simbolo.

Per Expert Advisors ed indicatori, è meglio utilizzare il modello di eventi di handling. Se durante l'handling di eventi OnTick() o OnCalculate(), la ricezione di dati per le serie temporali richieste è fallita, è necessario uscire dall'event handler, facendo affidamento sulla disponibilità di accesso durante la chiamata successiva dell'handler.

Esempio di script per l'Aggiunta di Storico

Consideriamo l'esempio di uno script che esegue una richiesta per ricevere lo storico per il simbolo selezionato da un trade server. Lo script è destinato all'esecuzione in un grafico di un simbolo selezionato; non importa il timeframe, perché, come è stato detto sopra, i dati sui prezzi sono ricevuti da un trade server come dei dati confezionati da 1 minuto, da cui tutte le timeseriespredefiniti, vengono costruite poi .

Scriviamo tutte le azioni riguardanti la ricezione di dati come una funzione separata CheckLoadHistory (simbolo, timeframe , start_date):

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

La funzione CheckLoadHistory() è concepita come una funzione universale che può essere chiamata da qualsiasi programma (Expert Advisor, Script o Indicatore), e pertanto richiede tre parametri di input: nome simbolo, periodo e la data di inizio per indicare l'inizio dello storico prezzi di cui si bisogno.

Inserire i necessari controlli nel codice della funzione prima di richiedere la storia mancante. Prima di tutto, dobbiamo fare in modo che il nome del simbolo ed il valore del periodo siano corretti:

```
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
```

Quindi facciamo in modo che il simbolo sia disponibile nella finestra di MarketWatch, vale a dire, la storia del simbolo sarà disponibile quando si invia una richiesta ad un trade server. Se non vi è un tale simbolo in MarketWatch, aggiungerlo utilizzando la funzione SymbolSelect().

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {f
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

Ora dobbiamo ricevere la data di inizio dello storico disponibile per la coppia simbolo/periodo indicata. Forse, il valore del parametro di input startdate, passato a CheckLoadHistory(), è all'interno dello storico disponibile; quindi la richiesta ad un trade server non è necessaria. Per ottenere la prima data per il simbolo-periodo del momento, viene utilizzata la funzione SeriesInfoInteger() con il modificatore SERIES_FIRSTDATE.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

Il prossimo importante controllo sta controllando il tipo di programma, da cui la funzione viene chiamata. Nota, inviando la richiesta per aggiornare le timeseries con lo stesso periodo di quello dell'indicatore, che chiama l'aggiornamento, è indesiderabile. La non desiderabilità della richiesta dei dati sullo stesso simbolo-periodo come quello dell'indicatore, è condizionata dal fatto che l'aggiornamento dei dati storici viene eseguito nello stesso thread in cui opera l'indicatore. Quindi, la possibilità di insorgenza di un punto morto è alta. Per controllare ciò, usare la funzione MQL5InfoInteger() con il modificatore MQL5_PROGRAM_TYPE.

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
      return(-4);
```

Se tutti i controlli hanno dato risultati positivi, fa l'ultimo tentativo senza fare riferimento al server commercio. Prima cerchiamo di scoprire la data di inizio, per i quali sono disponibili dati minute in formato HCC. Richiediamo questo valore utilizzando la funzione SeriesInfoInteger() con il modificatore SERIES_TERMINAL_FIRSTDATE e ancora confrontandolo con il valore del parametro start_date.

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- ci sono dati caricati per costruire timeseries
      if(first_date>0)
        {
         //--- forza la costruzione di timeseries
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- controlla data
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
```

Se dopo tutti i controlli il thread di esecuzione è ancora nel corpo della funzione CheckLoadHistory(), significa che vi è una necessità di richiedere i dati mancanti dei prezzi da un trade server. In primo luogo, viene restituito il valore di "Max barre nel chart" utilizzando la funzione TerminalInfoInteger():

```
  int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Avremo bisogno per impedire la richiesta di dati aggiuntivi. Poi trovare la prima vera data nello storico del simbolo sul trade server (indipendentemente dal periodo) utilizzando la già nota funzione SeriesInfoInteger() con il modificatore SERIES_SERVER_FIRSTDATE.

```
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
```

Dal momento che la richiesta è un'operazione asincrona, la funzione viene richiamata nel ciclo con un piccolo ritardo di 5 millisecondi fino a quando la variabile first_server_date riceve un valore, o l'esecuzione del ciclo viene interrotta da un utente (IsStopped() restituirà *true* in questo caso). Indichiamo un valore corretto della data di inizio, a partire dalla quale richiediamo i dati sui prezzi da un trade server.

```
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Attenzione: prima data server ",first_server_date," for ",
symbol," non corrisponde alla prima data della serie",first_date);
```

Se la data di inizio *first_server_date* del server è inferiore alla data di inizio *first_date* del simbolo in formato HCC, la voce corrispondente viene emessa nel Journal.

Ora siamo pronti a fare una richiesta ad un trade server per chiedere dati mancanti di prezzi. Effettuare la richiesta sotto forma di un ciclo ed iniziare a riempire il suo corpo:

```
   while(!IsStopped())
     {
      //1. attende la sincronizzazione tra le timeseries ri-costruite e lo storico int
      //2. ricevee il numero corrente di barre in questa serie storica
```

```
      //    se le barre sono più larghe di Max_bars_in_chart, possiamo uscire, il lavo
      //3. ottiene la data di inizio first_date in una timeseries ricostruita e la co
      //    se first_date è più piccolo di start_date, possiamo uscire, il lavoro è s
      //4. richiede dal server una nuova parte di storico - 100 bars iniziano dall'ult
     }
```

I primi tre punti sono attuati tramite i mezzi già noti.

```
   while(!IsStopped())
     {
      //--- 1.attende finchè il processo di ricostruzione della timeseries viene compl
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 2.richiede ora quante barre abbiamo
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         //--- barre più di quelle che possono essere disegnate nel chart, uscita
         if(bars>=max_bars) return(-2);
         //--- 3. restituisce la corrente data di inizio nella timeseries
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // la data di inizio era prima di quella richiesta, task completato
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //4. Richiede da un server una nuova parte dello storico - 100 barre a partire
     }
```

L'ultimo quarto punto viene lasciato - richiesta storico. Non possiamo fare riferimento ad un server direttamente, ma qualsiasi funzione-Copy inizia automaticamente la richiesta di invio ad un server, se lo storico in formato HCC non è sufficiente. Siccome il tempo della primissima data di inizio nella variabile *first_date* è il criterio semplice e naturale per valutare il grado richiesta di esecuzione, allora il modo ancor più semplice è quello di utilizzare la funzione CopyTime().

Quando si chiamano funzioni che copiano i dati provenienti da timeseries, si deve rilevare che il parametro *start* (numero della barra, a partire dal quale i dati di prezzo dovrebbero essere copiati) deve essere sempre all'interno dello storico disponibile del terminale. Se si hanno solo 100 barre, è privo di senso provare a copiare 300 barre partendo dalla barra con l'indice 500. Tale richiesta sarà intesa come erronea e non sarà trattata, cioè nessuna storia supplementare sarà caricata da un trade server.

Ecco perché si copierà dala barra 100 a partire dalla barra con indice *bars* (l'index). Ciò fornirà il corretto caricamento dello storico mancante, da un trade server. In realtà verrà caricato un po' di più rispetto alle 100 barre richieste, mentre il server invia lo storico di dimensioni più grandi.

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

Dopo l'operazione di copia, dobbiamo analizzare il numero di elementi copiati. Se il tentativo fallisce, allora il valore di *copied* sarà uguale a null ed il valore del contatore fail_cnt sarà aumentato di 1. Dopo 100 tentativi di fallimento, l'operazione della funzione viene interrotta.

```
int fail_cnt=0;
...
```

```
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- controllo dati
      if(times[0]<=start_date)  return(0);  // il valore copiato è più piccolo,
      if(bars+copied>=max_bars) return(-2); // le barre pronte sono più di quelle che
      fail_cnt=0;
     }
   else
     {
      //--- non più di 100 tentativi di fallimento in successo
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

Così, non solo viene implementato nella funzione il corretto handling della situazione attuale in qualsiasi fase di esecuzione, ma viene restituito anche il codice di terminazione, che può essere gestito dopo la chiamata della funzione CheckLoadHistory() per ottenere ulteriori informazioni. Ad esempio, in questo modo:

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Simbolo sconosciuto",InpLoadedSymbol);                    brea
      case -2 : Print("Richieste più barre di quelle che possono essere disegnate nel
      case -3 : Print("Esecuzione fermata dall'utente");                         brea
      case -4 : Print("L'indicatore non deve caricare i suoi propri dati");
      case -5 : Print("Caricamento fallito");                                    brea
      case  0 : Print("Tutti i dati caricati");                                   bre
      case  1 : Print("I dati già disponibili nella timeseries sono sufficienti");
      case  2 : Print("La timeseries è costruita da dati disponibili sul terminale");
      default : Print("L'esecuzione risulta non definita");
     }
```

Il codice completo della funzione può essere trovato nel esempio di uno script che mostra la corretta organizzazione di accesso ai dati con l'handling dei risultati della richiesta.

Codice:

```
//+------------------------------------------------------------------------------+
//|                                                      TestLoadHistory.mq5 |
//|                                  Copyright 2009, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- parametri di input
```

```
input string          InpLoadedSymbol="NZDUSD";    // Simbolo da caricare
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;   // Periodo da caricare
input datetime        InpStartDate=D'2006.01.01'; // Data di inizio
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   Print("Start load",InpLoadedSymbol+","+GetPeriodName(InpLoadedPeriod),"from",InpSta
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Simbolo sconosciuto ",InpLoadedSymbol);             break;
      case -2 : Print("Richieste più barre delle massime barre (max bars) nel chart");
      case -3 : Print("Il progamma è stato fermato");                       break;
      case -4 : Print("L'indicatore non dovrebbe caricare i suoi stessi dati");      b
      case -5 : Print("Caricamento fallito");                               break;
      case  0 : Print("Caricamento OK");                                  break;
      case  1 : Print("Caricato precedentemente");                          break;
      case  2 : Print("Caricato precedentemente e costruito");               break;
      default : Print("Risultato sconosciuto");
     }
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("First date ",first_date," - ",bars," bars");
//---
  }
//+------------------------------------------------------------------------------+
//|                                                                              |
//+------------------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
//--- controlla symbol & period
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- controlla se il simbolo è selezionato nel MarketWatch
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- controllare se sono presenti dati
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

```
//--- non chiedere per il caricamento dei propri dati se si tratta di un indicatore
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sym
      return(-4);
//--- secondo tentativo
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- ci sono dati caricati per costruire timeseries
      if(first_date>0)
        {
         //--- forza la costruzione di timeseries
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- controlla data
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
//--- Massime barre nel chart dalle opzioni del terminale
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- Carica info storiche dei simboli
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- Fissa la data di inizio per il caricamento
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Avviso: prima data del server",first_server_date," for ",symbol,
            " non corrisponde alla data della prima serie ",first_date);
//--- Carica dati passo per passo
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- attendi per la costruzione di timeseries
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- chiedi per barre costruite
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- chiedi per la prima data
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- la copia della prossima parte, forza il caricamento dei dati
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- controlla i dati
         if(times[0]<=start_date)  return(0);
```

```
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- non più di 100 tentativi di fallimento
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- fermato
   return(-3);
  }
//+------------------------------------------------------------------------------+
//| Restituisce i valori stringa del periodo                                     |
//+------------------------------------------------------------------------------+
string GetPeriodName(ENUM_TIMEFRAMES period)
  {
   if(period==PERIOD_CURRENT) period=Period();
//---
   switch(period)
     {
      case PERIOD_M1:  return("M1");
      case PERIOD_M2:  return("M2");
      case PERIOD_M3:  return("M3");
      case PERIOD_M4:  return("M4");
      case PERIOD_M5:  return("M5");
      case PERIOD_M6:  return("M6");
      case PERIOD_M10: return("M10");
      case PERIOD_M12: return("M12");
      case PERIOD_M15: return("M15");
      case PERIOD_M20: return("M20");
      case PERIOD_M30: return("M30");
      case PERIOD_H1:  return("H1");
      case PERIOD_H2:  return("H2");
      case PERIOD_H3:  return("H3");
      case PERIOD_H4:  return("H4");
      case PERIOD_H6:  return("H6");
      case PERIOD_H8:  return("H8");
      case PERIOD_H12: return("H12");
      case PERIOD_D1:  return("Daily");
      case PERIOD_W1:  return("Weekly");
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("periodo sconosciuto");
  }
```

SeriesInfoInteger

Restituisce informazioni sullo stato dei dati storici. Ci sono 2 varianti di chiamate di funzione.

Restituisce direttamente il valore della proprietà.

```
long  SeriesInfoInteger(
   string                      symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES             timeframe,        // periodo
   ENUM_SERIES_INFO_INTEGER    prop_id,          // identificatore proprietà
   );
```

Restituisce true o false a seconda del successo dell'esecuzione della funzione.

```
bool  SeriesInfoInteger(
   string                      symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES             timeframe,        // periodo
   ENUM_SERIES_INFO_INTEGER    prop_id,          // ID proprietà
   long&                       long_var          // variabile per ottenere informazioni
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

prop_id

[in] Identificatore della richiesta di proprietà, il valore dell' enumerazione ENUM_SERIES_INFO_INTEGER.

long_var

[out] Variabile per cui viene piazzato il valore della proprietà richiesta.

Valore restituito

Nel primo caso, restituisce il valore di tipo tipo long.

Per il secondo caso, restituisce true, se la proprietà specificata è disponibile ed il suo valore è stato messo nella variabile long_var, altrimenti restituisce false. Per ulteriori informazioni sull' errore, chiamare GetLastError().

Esempio:

```
voidOnStart()
  {
//---
   Print("Numero totale di barre per il periodo-simbolo specificati al momento = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("La prima data per il simbolo-periodo in questo momento = ",
```

```
        (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));

  Print("La prima data nello storico per il simbolo-periodo sul server = ",
        (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

  Print("I dati dei simboli sono sincronizzati = ",
        (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
 }
```

Bars

Restituisce il numero di barre contate nello storico di un simbolo e periodo specificati. Ci sono 2 varianti di chiamate funzioni.

Richiedi tutte le barre dello storico

```
int  Bars(
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe         // periodo
   );
```

Richiede le barre dello storico per l'intervallo di tempo selezionato

```
int  Bars(
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data e tempo di inizio
   datetime         stop_time         // data ed ora di fine
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_time

[in] Orario barra corrispondente al primo elemento.

stop_time

[in] Orario barra corrispondente all'ultimo elemento.

Valore restituito

Se i parametri start_time e stop_time sono definiti, la funzione restituisce il numero di barre nell'intervallo di tempo specificato, in caso contrario restituisce il numero totale di barre.

Nota

Se i dati per le timeseries con parametri specificati non sono formati nel terminale al momento della chiamata di funzione Bars(), o i dati delle timeseries non sono sincronizzati con un trade server al momento della chiamata di funzione, la funzione restituisce un valore pari a zero.

Quando si richiede il numero di barre in un intervallo di tempo specificato, solo le barre con un tempo aperto che rientrano nell'intervallo vengono considerate. Ad esempio, se l'attuale giorno della settimana è Sabato e la richiesta è fatta per il numero di W1 barre con start_time=last_tuesday e stop_time=last_friday, la funzione restituirà 0 poiché il tempo di apertura di un lasso di tempo W1 è sempre Domenica e non una singola barra W1 rientra nell'intervallo specificato.

Richiesta di esempio per il numero di tutte le barre dello storico:

```
int bars=Bars(_Symbol,_Period);
if(bars>0)
  {
   Print("Numero di barre nello storico del terminale per il simbolo-periodo al mor
  }
else  //non ci sono barre disponibili
  {
   //--- i dati sul simbolo possono non essere sincronizzati con i dati sul server
   bool synchronized=false;
   //--- contatore del ciclo
   int attempts=0;
   // fa 5 tentativi di attesa per la sincronizzazione
   while(attempts<5)
     {
      if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
        {
         //--- sincronizzazione eseguita, uscita
         synchronized=true;
         break;
        }
      //--- incrementa il contatore
      attempts++;
      //--- attende 10 millisecondi fino alla prossima iterazione
      Sleep(10);
     }
   //--- esce dal loop dopo la sincronizzazione
   if(synchronized)
     {
      Print("Numero di barre nello storico del terminale per il simbolo-periodo, al
      Print("La prima data nello storico del terminale per il simbolo-periodo al mo
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
      Print("La prima data nello storico per il simbolo sul server = ",
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
     }
   //--- la sincronizzazione dei dati non è avvenuta
   else
     {
      Print("Fallimento nell'ottenere il numero di barre per ",_Symbol);
     }
  }
```

Richiesta di campione per il numero di barre nell'intervallo specificato:

```
int n;
datetime date1 = D'2016.09.02 23:55'; // Friday
datetime date2 = D'2016.09.05 00:00'; // Monday
datetime date3 = D'2016.09.08 00:00'; // Thursday
//---
n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
```

```
   Print("Numero di barre: ",n); // Output: "Numero di barre: 8", la barra H2 è consid
   n=Bars(_Symbol,PERIOD_D1,date1,date2);
   Print("Numero di barre: ",n); // Output: "Numero di barre: 1", giacchè una barra ap
   n=Bars(_Symbol,PERIOD_W1,date2,date3);
   Print("Numero di barre: ",n); // Output: "Numero di barre: 0", giacchè nemmeno una
```

Vedi anche

Funzioni di elaborazione di Eventi

BarsCalculated

Restituisce il numero di dati calcolati per l'indicatore specificato.

```
int  BarsCalculated(
   int       indicator_handle,     // handle indicatore
   );
```

Parametri

indicator_handle

[in] L' handle indicatore, restituito dalla funzione indicatore corrispondente.

Valore restituito

Restituisce la quantità di dati calcolati nel buffer indicatore o -1 in caso di errore (dati non ancora calcolati)

Nota

La funzione è utile quando è necessario ottenere i dati degli indicatori immediatamente dopo la loro creazione (l'handle indicatore diventa disponibile).

Esempio:

```
voidOnStart()
  {
   double Ups[];
//--- imposta l'ordine timeseries per gli array
   ArraySetAsSeries(Ups,true);
//--- crea l'handle per l'indicatore Fractal
   int FractalsHandle=iFractals(NULL,0);
//--- resetta il codice errore
   ResetLastError();
//--- prova a copiare i valori degli indicatori
   int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Sleep(50);
      for(i=0;i<100;i++)
        {
         if(BarsCalculated(FractalsHandle)>0)
            break;
         Sleep(50);
        }
      copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
      if(copied<=0)
        {
         Print("Fallimento nel copiare i frattali superiori. Error = ",GetLastError(),
         "i = ",i,"    copied = ",copied);
         return;
        }
```

```
      else
        Print("Frattali superiori copiati",
        "i = ",i,"    copied = ",copied);
    }
  else Print("Frattali superiori copiati. ArraySize = ",ArraySize(Ups));
}
```

IndicatorCreate

La funzione restituisce l'handle di un indicatore tecnico specificato creato in base all'array di parametri di MqlParam tipo.

```
int  IndicatorCreate(
   string           symbol,                    // nome simbolo
   ENUM_TIMEFRAMES  period,                    // timeframe
   ENUM_INDICATOR   indicator_type,            // tipo indicatore per l'enume
   int              parameters_cnt=0,          // numero di parametri
   const MqlParam&  parameters_array[]=NULL,   // array di parametri
   );
```

Parametri

symbol

[in] Nome di un simbolo, dei dati di cui viene calcolato l'indicatore. NULL significa il corrente simbolo

period

[in] Il valore del periodo può essere uno dei valori dell' enumerazione ENUM_TIMEFRAMES, 0 significa il timeframe corrente.

indicator_type

[in] Tipo indicatore, può essere uno dei valori dell' enumerazione ENUM_INDICATOR.

parameters_cnt

[in] Il numero di parametri passati nell' array parameters_array[]. Gli elementi dell' array ha un particolare tipo struttura MqlParam. Per impostazione predefinita, zero parametri - non vengono passati. Se si specifica un numero di parametri diverso da zero, il parametro *parameters_array* è obbligatorio. È possibile passare non più di 64 parametri.

parameters_array[]=NULL

[in] Un array di tipo MqlParam, i cui elementi contengono il tipo ed il valore di ciascun parametro di input di un indicatore tecnico.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE.

Nota

Se viene creato l'handle indicatore di tipo IND_CUSTOM, il campo *tipo* del primo elemento dell'array di parametri di input *parameters_array* deve avere il valore dell'enumerazione TYPE_STRINGENUM_DATATYPE, ed il campo *valore_stringa* del primo elemento deve contenere il nome dell'indicatore personalizzato. L'indicatore personalizzato deve essere compilato (file con estensione EX5) e localizzato nella directory MQL5/Indicators del terminale client o in una sottodirectory.

Indicatori che richiedono il testing vengono definiti automaticamente dalla chiamata della funzione iCustom(), se il parametro corrispondente è impostato attraverso la costante stringa. Per tutti gli

altri casi (utilizzo della funzione IndicatorCreate() o uso di una stringa non-costante nel parametro che imposta il nome indicatore) la proprietà #property tester_indicator è necessaria:

```
#property tester_indicator "indicator_name.ex5"
```

Se la prima forma di chiamata viene usata nell'indicatore personalizzato, è possibile inoltre indicare come ultimo parametro quali dati vengono calcolati quando si passano i parametri di input. Se il parametro "Applica a" non viene indicato, il calcolo predefinito è basato su valori PRICE_CLOSE.

Esempio:

```
voidOnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- crea iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);
   ArrayResize(params,4);
//--- imposta ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- imposta ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- imposta ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- imposta applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- crea MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- crea iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
   ArrayResize(params,4);
//--- imposta fast ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- imposta slow ma_period
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- imposta periodo smooth for difference
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- imposta l'handle indicatore come applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- crea MACD basato su moving average
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- usa l'indicatore
//--- . . .
//--- rilascia indicatore (first h_MACD)
```

```
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

IndicatorParameters

Basato sull' handle specificato, restituisce il numero di parametri di input dell'indicatore, nonché i valori ed i tipi di parametri.

```
int  IndicatorParameters(
   int               indicator_handle,     // handle indicatore
   ENUM_INDICATOR&   indicator_type,       // la variable per la ricezione del tipo di
   MqlParam&         parameters[]          // l' array per la ricezione dei parametri
   );
```

Parametri

indicator_handle

[in] L' handle dell'indicatore, per cui è necessario conoscere il numero di parametri su cui è calcolato.

indicator_type

[out] A variable if the ENUM_INDICATOR tipo, con il quale l'indicatore verrà scritto.

parameters[]

[out] Un array dinamico per ricevere valori di tipo MqlParam, in cui verrà scritto l'elenco dei parametri indicatori. La grandezza dell'array viene restituita dalle funzione IndicatorParameters().

Valore restituito

Il numero di parametri di input dell'indicatore con l'handle specificato. In caso di un errore restituisce -1. Per ulteriori informazioni sull'errore chiamare la funzione GetLastError().

Esempio:

```
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {

//--- Il numero di finestre sul grafico (almeno una finestra principale è sempre prese
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- Va attraverso le finestre grafico
   for(int w=0;w<windows;w++)
     {
      // --- Il numero di indicatori in questa finestra/sottofinestra
      int total=ChartIndicatorsTotal(0,w);
      //--- Prende tutti gli indicatori nella finestra
      for(int i=0;i<total;i++)
        {
         //--- Ottiene il nome breve dell'indicatore
         string name=ChartIndicatorName(0,w,i);
         //--- Ottiene l'handle indicatore
         int handle=ChartIndicatorGet(0,w,name);
         //--- Aggiunge al log
```

```
         PrintFormat("Window=%d,  indicator #%d,  handle=%d",w,i,handle);
         //---
         MqlParam parameters[];
         ENUM_INDICATOR indicator_type;
         int params=IndicatorParameters(handle,indicator_type,parameters);
         //--- La testata del messaggio
         string par_info="Short name "+name+", type "
                         +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
         //---
         for(int p=0;p<params;p++)
           {
            par_info+=StringFormat("parameter %d: type=%s, long_value=%d, double_value
                                   p,
                                   EnumToString((ENUM_DATATYPE)parameters[p].type),
                                   parameters[p].integer_value,
                                   parameters[p].double_value,
                                   parameters[p].string_value
                                   );
           }
         Print(par_info);
        }
      //--- Fatto per tutti gli indicatori nella finestra
     }
//---
  }
```

Vedi anche

ChartIndicatorGet()

IndicatorRelease

La funzione rimuove un handle indicatore e rilascia il blocco di calcolo dell'indicatore, se non è usato da nessun altro.

```
bool  IndicatorRelease(
   int       indicator_handle     // handle indicatore
   );
```

Valore restituito

Restituisce true in caso di successo, altrimenti restituisce false.

Nota

La funzione permette di rimuovere un handle indicatore, se è non è più necessario, risparmiando così memoria. L' handle viene rimosso immediatamente, il blocco di calcolo viene eliminato in un certo tempo (se non è più chiamato).

Quando si lavora nello strategy tester, la funzione IndicatorRelease() non viene eseguita.

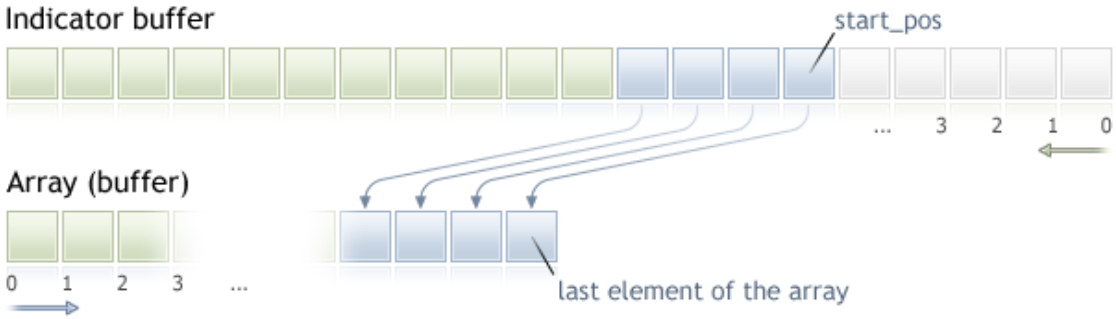
Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                Test_IndicatorRelease.mq5 |
//|                                Copyright 2010, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parametri di input
input int                MA_Period=15;
input int                MA_shift=0;
input ENUM_MA_METHOD     MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- salverà l'handle indicatore
int MA_handle=INVALID_HANDLE;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- crea l'handle indicatore
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- elimina la variabile globale
   if(GlobalVariableCheck("MA_value"))
      GlobalVariableDel("MA_value");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione tick dell'Expert                                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- se il valore della variabile globale non esiste
   if(!GlobalVariableCheck("MA_value"))
     {
      //--- ottiene il valore dell'indicatore nelle ultime due barre
      if(MA_handle!=INVALID_HANDLE)
        {
         //--- array dinamico per i valori degli indicatori
         double values[];
         if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
           {
            //--- ricorda il valore della variabile globale sulla penultima barra
            if(GlobalVariableSet("MA_value",values[0]))
              {
               //--- libera l'handle dell'indicatore
               if(!IndicatorRelease(MA_handle))
                  Print("IndicatorRelease() fallito. Error ",GetLastError());
               else MA_handle=INVALID_HANDLE;
              }
            else
               Print("GlobalVariableSet fallito. Error ",GetLastError());
           }
        }
     }
//---
  }
```

CopyBuffer

Ottiene dati di un buffer specificato di un certo indicatore nella quantità necessaria.



Il conteggio degli elementi di dati copiati (buffer indicatore con l'indice buffer_num) dalla posizione iniziale viene eseguito dal presente al passato, cioè, la posizione iniziale di 0 significa la barra di corrente (valore indicatore per la barra corrente).

Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare un array dinamico come un buffer[] recipient buffer, poiché la funzione CopyBuffer() tenta di allocare la dimensione dell'array di ricezione per la dimensione dei dati copiati. Se un buffer di indicatore (array che è pre-assegnato per memorizzare i valori degli indicatori dalla funzione SetIndexBufer()) viene usato come buffer[] recipient array, la copia parziale è consentita. Un esempio può essere trovato nell' indicatore personalizzato Awesome_Oscillator.mql5 nel pacchetto standard del terminale .

Se avete bisogno di fare una copia parziale dei valori degli indicatori in un altro array (buffer non-indicatore), è necessario utilizzare un array intermedio, al quale il numero desiderato viene copiato. Dopo ciò, condurre la copia dell'elemento del numero richiesto di valori nel posto richiesto se un gruppo ricevente da questo intermedia.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyBuffer(
   int       indicator_handle,     // handle indicatore
   int       buffer_num,           // numero del buffer indicatore
   int       start_pos,            // posizione di inizio
   int       count,                // quantità da copiare
   double    buffer[]              // array target in cui copiare
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyBuffer(
   int       indicator_handle,     // handle indicatore
```

```
   int       buffer_num,          // numero del buffer indicatore
   datetime  start_time,          // ora e data di inizio
   int       count,               // quantità da copiare
   double    buffer[]             // array destinazione da copiare
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyBuffer(
   int       indicator_handle,    // handle indicatore
   int       buffer_num,          // numero del buffer indicatore
   datetime  start_time,          // ora e data di inizio
   datetime  stop_time,           // ora e data di fine
   double    buffer[]             // array target in cui copiare
   );
```

Parametri

indicator_handle

[in] L' handle indicatore, restituito dalla funzione indicatore corrispondente.

buffer_num

[in] In numero di buffer indicatore.

start_pos

[in] La posizione del primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] Tempo Barra, corrispondente al primo elemento.

stop_time

[in] Tempo Barra, corrispondente all'ultimo elemento.

buffer[]

[out] Array of double type.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso di errore.

Nota

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta dal momento dell'espirazione del timeout.

Esempio:

```
//+------------------------------------------------------------------------------------+
//|                                                                 TestCopyBuffer3.mq5 |
//|                                        Copyright 2009, MetaQuotes Software Corp. |
//|                                                                 https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- buffers indicatore
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Parameter AsSeries = ",AsSeries);
   Print("Buffer Indicatore dopo SetIndexBuffer() è una = ",
         ArrayGetAsSeries(MABuffer));
//--- Imposta il nome breve indicatore
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- imposta AsSeries (dependes dal parametro di input)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Buffer indicatore dopo ArraySetAsSeries(MABuffer,true); è una timeseries = "
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
```

```
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- controllare se tutti i dati calcolati
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- non possiamo copiarli tutti
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- l'ultimo valore viene sempre copiato
      to_copy++;
     }
//--- prova a copiare
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

L'esempio precedente illustra come un buffer indicatore viene riempito con i valori di un altro buffer indicatore, dall'indicatore sullo stesso simbolo/periodo.

Vedere l' esempio dettagliato della richiesta dati dello storico di cui al punto Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

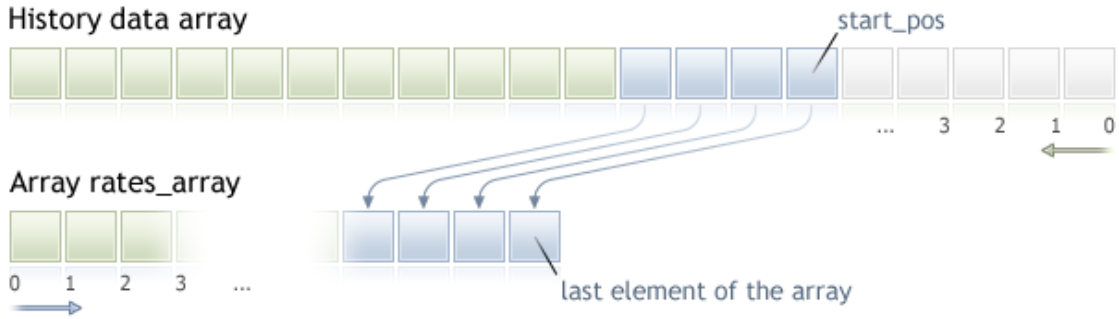
- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

Vedi anche

Proprietà degli Indicatori Personalizzati, SetIndexBuffer

CopyRates

Ottiene i dati storici della struttura MqlRates di un determinat simbolo-periodo nella quantità specificata nell' array rates_array. L'ordinamento degli elementi dei dati copiati è dal presente al passato, vale a dire, la posizione di partenza di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyRates(
   string           symbol_name,       // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   int              start_pos,         // posizione inizio
   int              count,             // conteggio dati da copiare
   MqlRates         rates_array[]      // array target da copiare
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyRates(
   string           symbol_name,       // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   datetime         start_time,        // data ed ora di inizio
   int              count,             // conteggio dati da copiare
   MqlRates         rates_array[]      // array target da copiare
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyRates(
```

```
   string           symbol_name,       // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   datetime         start_time,        // data ed ora di inizio
   datetime         stop_time,         // data ed ora di fine
   MqlRates         rates_array[]      // array target da copiare
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_time

[in] Orario barra per il primo elemento da copiare.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare.

rates_array[]

[out] Array di MqlRates type.

Valore restituito

Restituisce il numero di elementi copiati o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Esempio:

```
voidOnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Barre copiate: "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("Fallimento nell'ottenimento dei dati dello storico per il simbolo ",Sy
  }
```

Vedere un esempio dettagliato di richiesta dati storici di cui al punto Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,

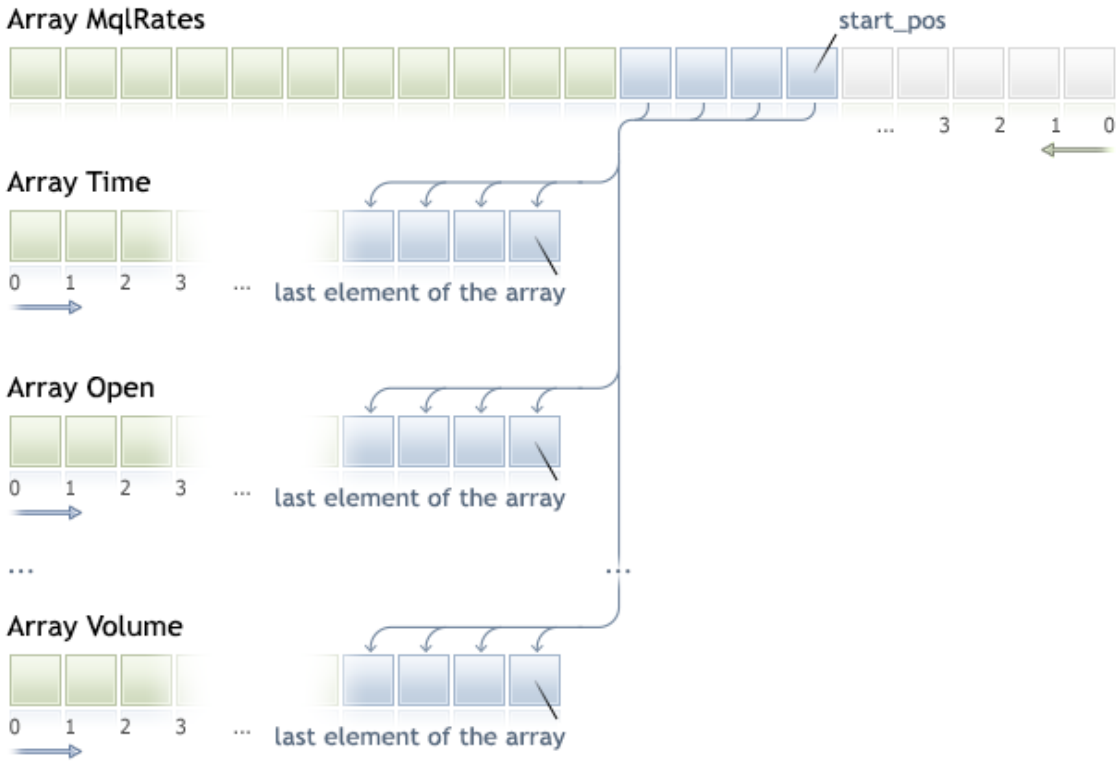
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

Vedi anche

Strutture e Classi,TimeToString, StringFormat

CopySeries

Ottiene le Timeseries sincronizzate dalla struttura MqlRates per il simbolo-periodo specificato e la quantità specificata. I dati vengono ricevuti nella serie di array indicati. Gli elementi sono conteggiati dal presente al passato, il che significa che la posizione di partenza uguale a 0 significa la barra corrente.



Se la quantità di dati da copiare è sconosciuta, si consiglia di utilizzare gli array dinamici come array riceventi, poichè se la quantità di dati eccede ciò che un array può contenere, questo può causare il tentativo di ridistribuire l'array per adattarsi a tutti i dati richiesti.

Se si ha la necessità di copiare una quantità predeterminata di dati, si consiglia di utilizzare un buffer allocato staticamente per evitare riallocazioni di memoria inutili.

La proprietà dell'array ricevente — as_series=true or as_series=false — verrà ignorato: durante la copia, l'elemento Timeseries più vecchio verrà copiato all'inizio della memoria fisica allocata per l'array.

```
int  CopySeries(
   string           symbol_name,       // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   int              start_pos,         // posizione iniziale
   int              count,             // quantità da copiare
   ulong            rates_mask,        // combinazione di flag per specificare la seri
   void&            array1[],          // array per ricevere i dati della prima Timese
   void&            array2[]           // array per ricevere i dati della seconda Time
   ...
```

```
   );
```

Parametri

symbol_name

[in] Simbolo.

timeframe

[in] Periodo.

start_pos

[in] Indice del primo elemento copiato.

count

[in] Numero di elementi copiati.

rates_mask

[in] Una Combinazione di flag dell'enumerazione ENUM_COPY_RATES.

array1, array2,...

[out] Array del tipo appropriato per ricevere le Timeseries dalla struttura MqlRates. L'ordine degli array passati alla funzione deve corrispondere all'ordine dei campi nella struttura MqlRates.

Valore Restituito

Il numero degli elementi copiati o -1 in caso di errore.

Nota

Se l'intero intervallo dei dati richiesti è fuori dai dati disponibili sul server, la funzione restituisce -1. Se i dati richiesti sono oltre TERMINAL_MAXBARS (il numero massimo di barre sul grafico), la funzione restituisce ugualmente -1.

Quando si richiedono dati da un indicatore, la funzione restituisce immediatamente -1 se le Timeseries richieste non sono ancora costruite o devono essere scaricate dal server. Tuttavia, questo avvierà il download/costruzione dei dati.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server se il terminale non dispone dei dati appropriati localmente, o la costruzione delle timeseries necessarie inizia se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che è pronta nel momento in cui scade il timeout, tuttavia il download dello storico continua e la funzione restituisce più dati durante la successiva identica richiesta.

Differenza tra CopySeries e CopyRates

La funzione CopySeries consente di ottenere solo le timeseries necessarie in diversi array specificati durante una chiamata, mentre tutti i dati timeseries saranno sincronizzati. Ciò significa che tutti i valori negli array risultanti ad un determinato indice N apparterranno alla stessa barra sulla coppia Simbolo/Timeframe specificati. Pertanto, non è necessario che il programmatore si assicuri della sincronizzazione di tutte le Timeserie ricevute dall'orario di apertura della barra.

A differenza di CopyRates, che restituisce il set completo di timeseries come array MqlRates, la funzione CopySeries consente al programmatore di ottenere solo le timeseries richieste come array

separati. Questo può essere fatto specificando una combinazione di flag per selezionare il tipo di timeseries. L'ordine degli array passati alla funzione deve corrispondere all'ordine dei campi nella struttura MqlRates:

```
struct MqlRates
  {
   datetime time;         // periodo d'inizio
   double   open;         // prezzo di apertura
   double   high;         // prezzo massimo del periodo
   double   low;          // prezzo minimo del periodo
   double   close;        // prezzo di chiusura
   long     tick_volume;  // tick volume
   int      spread;       // spread
   long     real_volume;  // volume degli scambi
  }
```

Pertanto, se è necessario ottenere i valori delle timeseries di *time*, *close* e*real_volume* per le ultime 100 barre del Simbolo/Timeframe corrente, è necessario utilizzare la seguente chiamata:

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_VOLUME_REAL,time,c
```

Attenzione all'ordine degli array "*time*, *close*, *volume*" — deve corrispondere al l'ordine dei campi della struttura MqlRates. L'ordine dei valori nel *rates_mask* non ha importanza. La maschera potrebbe essere la seguente:

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

Esempio:

```
//--- parametri di input
input datetime InpDateFrom=D'2022.01.01 00:00:00';
input datetime InpDateTo  =D'2023.01.01 00:00:00';
input uint     InpCount   =20;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- array per ottenere timeseries dalla struttura dei prezzi MqlRates
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//--- richiesta dei prezzi di chiusura ad un array double
   ResetLastError();
   int res1=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE, time1, close);
```

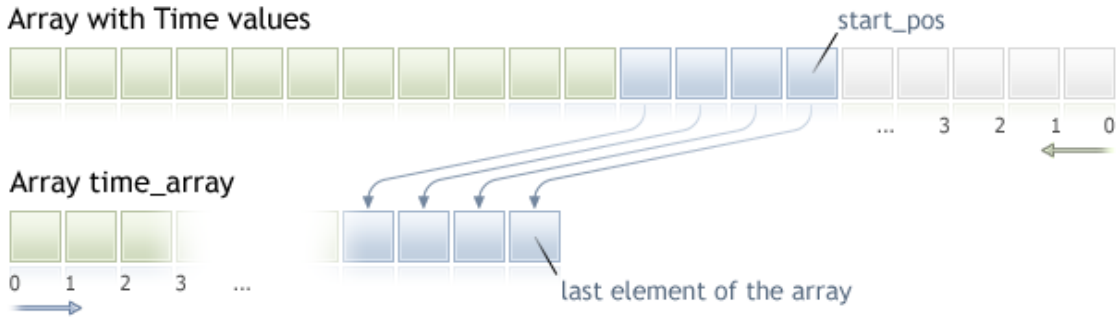
```
   PrintFormat("1. CopySeries  returns %d values. Error code=%d", res1, GetLastError()
   ArrayPrint(close);

//--- ora richiediamo anche i prezzi di apertura; utilizza array float per i prezzi di
   ResetLastError();
   int res2=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_OPEN, time2, open,
   PrintFormat("2. CopySeries  returns %d values. Error code=%d", res2, GetLastError()
   ArrayPrint(closef);
//--- Confrontare i dati ricevuti
   if((res1==res2) && (time1[0]==time2[0]))
     {
      Print("  | Time              |    Open      | Close double | Close float |");
      for(int i=0; i<10; i++)
        {
         PrintFormat("%d | %s |   %.5f    |   %.5f    |   %.5f   |",
                     i, TimeToString(time1[i]), open[i], close[i], closef[i]);
        }
     }
//--- Risultato
      1. CopySeries restituisce 20 valori. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
      2. CopySeries restituisce 20 valori. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
        | Time              |    Open      | Close double | Close float |
      0 | 2023.03.01 17:00 |   1.06660    |   1.06722    |   1.06722   |
      1 | 2023.03.01 18:00 |   1.06722    |   1.06733    |   1.06733   |
      2 | 2023.03.01 19:00 |   1.06734    |   1.06653    |   1.06653   |
      3 | 2023.03.01 20:00 |   1.06654    |   1.06520    |   1.06520   |
      4 | 2023.03.01 21:00 |   1.06520    |   1.06573    |   1.06573   |
      5 | 2023.03.01 22:00 |   1.06572    |   1.06649    |   1.06649   |
      6 | 2023.03.01 23:00 |   1.06649    |   1.06694    |   1.06694   |
      7 | 2023.03.02 00:00 |   1.06683    |   1.06675    |   1.06675   |
      8 | 2023.03.02 01:00 |   1.06675    |   1.06684    |   1.06684   |
      9 | 2023.03.02 02:00 |   1.06687    |   1.06604    |   1.06604   |
//---
  }
```

Vedere anche

Strutture e classi, CopyRates

CopyTime

La funzione ottiene in time_array i dati storici dei prezzi di apertura delle barre la coppia simbolo-periodo selezionati nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyTime(
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   int              start_pos,        // posizione inizio
   int              count,            // conteggio dati da copiare
   datetime         time_array[]      // target array to copy open times
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyTime(
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data e tempo di inizio
   int              count,            // conteggio dati da copiare
   datetime         time_array[]      // target array to copy  open times
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyTime(
```

```
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data e tempo di inizio
   datetime         stop_time,        // data ed ora di fine
   datetime         time_array[]      // target array to copy open times
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] L'orario di avviamento per il primo elemento da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare..

time_array[]

[out] Array di tipo datetime.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

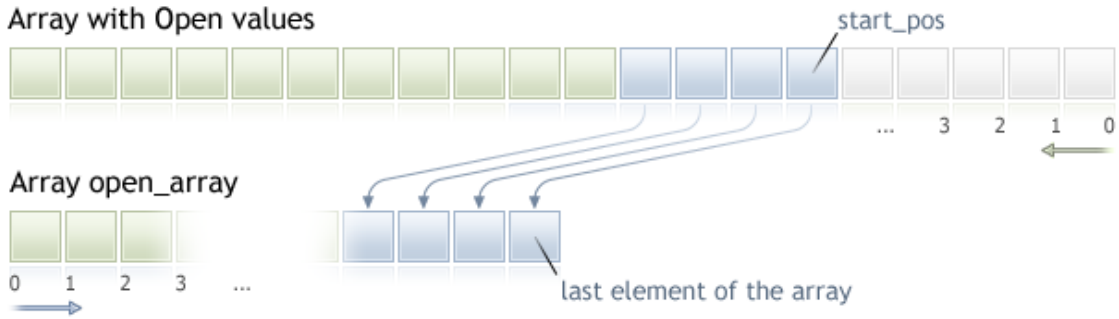
Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Vedere un esempio dettagliato di richiesta dati storici di cui al punto Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyOpen

La funzione ottiene in open_array i dati storici dei prezzi della barra open per la coppia simbolo-periodo selezionati nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyOpen(
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   int              start_pos,        // posizione inizio
   int              count,            // conteggio dati da copiare
   double           open_array[]      // array target in cui copiare i prezzi di apertu
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyOpen(
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data e tempo di inizio
   int              count,            // conteggio dati da copiare
   double           open_array[]      // array target per i prezzi di apertura delle ba
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyOpen(
```

```
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data e tempo di inizio
   datetime         stop_time,        // data ed ora di fine
   double           open_array[]      // array target per i prezzi di apertura dei valo
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] L'orario di avviamento per il primo elemento da copiare.

stop_time

[in] L'orario di inizio per l'ultimo elemento da copiare.

open_array[]

[out] Array of double type.

Valore restituito

Restituisce il numero di elementi nell'array o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

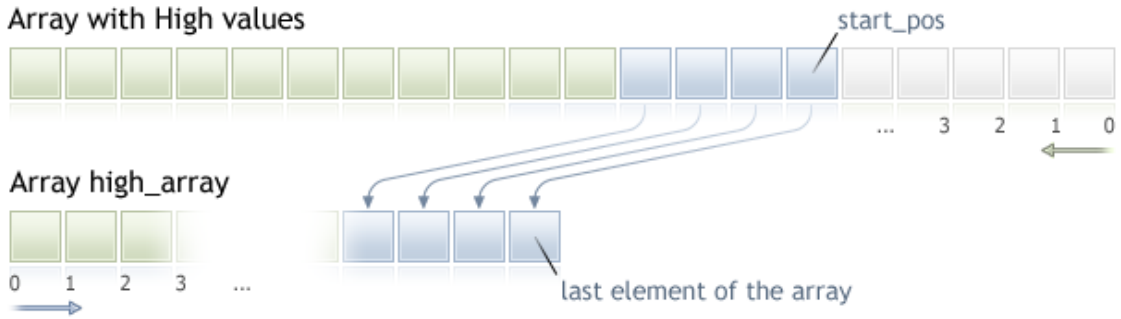
Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Vedere un esempio dettagliato di richiesta dati storici di cui al punto Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyHigh

La funzione ottiene in high_array i dati storici dei prezzi più alti della barra per il la coppia simbolo-periodo selezionato nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyHigh(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   int              start_pos,        // posizione di inizio
   int              count,            // conteggio dati da copiare
   double           high_array[]      // array target in cui copiare
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyHigh(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   int              count,            // conteggio dati da copiare
   double           high_array[]      // array target in cui copiare
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyHigh(
```

```
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   datetime         stop_time,        // data ed ora di fine
   double           high_array[]      // array target in cui copiare
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] L'orario di avviamento per il primo elemento da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare.

high_array[]

[out] Array of double type.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Esempio:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un esempio per l'output di High[i] and Low[i]"
#property description "per barre scelte a caso"

double High[],Low[];
//+------------------------------------------------------------------------------+
//| Ottiene Low per l'indice barra specificato                                   |
//+------------------------------------------------------------------------------+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double low=0;
   ArraySetAsSeries(Low,true);
   int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
   if(copied>0 && index<copied) low=Low[index];
   return(low);
  }
//+------------------------------------------------------------------------------+
//| Ottiene High per l'indice barra specificato                                  |
//+------------------------------------------------------------------------------+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double high=0;
   ArraySetAsSeries(High,true);
   int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
   if(copied>0 && index<copied) high=High[index];
   return(high);
  }
//+------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                    |
//+------------------------------------------------------------------------------+
void OnTick()
```

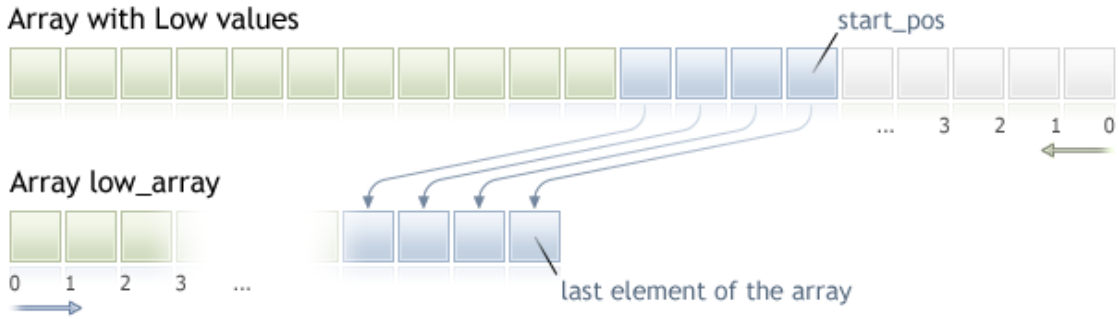
```
  {
//--- su ogni tick diamo in output i valori High e Low per la barra con l'indice,
//--- che è uguale al secondo, su quale il tick è arrivato
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
          sec,iLow(Symbol(),0,sec));
  }
```

Vedere un esempio dettagliato di dati storici richiedente nella Metodi di oggetti Binding sezione. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyLow

La funzione ottiene in low_array i dati storici dei prezzi più bassi della barra per il la coppia simbolo-periodo selezionato nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyLow(
   string            symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES   timeframe,        // periodo
   int               start_pos,        // posizione inizio
   int               count,            // conteggio dati da copiare
   double            low_array[]       // array target in cui copiare
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyLow(
   string            symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES   timeframe,        // periodo
   datetime          start_time,       // data e tempo di inizio
   int               count,            // conteggio dati da copiare
   double            low_array[]       // array target in cui copiare
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyLow(
```

```
   string           symbol_name,      // nome simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data e tempo di inizio
   datetime         stop_time,        // data ed ora di fine
   double           low_array[]       // array target in cui copiare
   );
```

Parametri

symbol_name

[in] Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] Tempo barra, corrispondente al primo elemento da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare.

low_array[]

[out] Array of double type.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Vedere un esempio dettagliato di richiesta dati storici di cui al punto Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

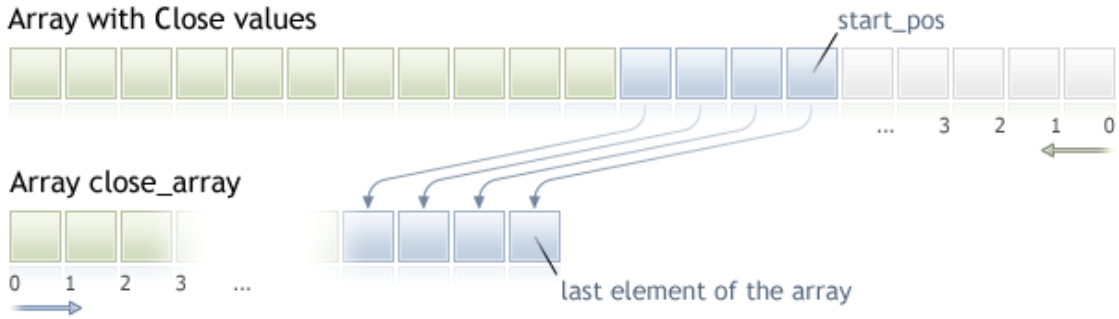
- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

Vedi anche

CopyHigh

CopyClose

La funzione ottiene in close_array i dati storici dei prezzi della barra close per la coppia simbolo-periodo selezionati nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyClose(
   string           symbol_name,       // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   int              start_pos,         // posizione inizio
   int              count,             // conteggio dati da copiare
   double           close_array[]      // array target in cui copiare
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyClose(
   string           symbol_name,       // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   datetime         start_time,        // data ed ora di inizio
   int              count,             // conteggio dati da copiare
   double           close_array[]      // array target in cui copiare
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyClose(
```

```
   string           symbol_name,       // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,         // periodo
   datetime         start_time,        // data ed ora di inizio
   datetime         stop_time,         // stop date and time
   double           close_array[]      // array target in cui copiare
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] L'orario di avviamento per il primo elemento da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare.

close_array[]

[out] Array of double type.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono i dati da un Expert Advisor o da uno script, verrà avviato il downloading dal server, se il terminale non dispone di questi dati a livello locale, o inizierà la costruzione di una timeseries richiesta, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati dalla data di inizio ed il numero di elementi richiesti, solo i dati con data minore (prima) o uguale alla data specificata vengono restituiti. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i

prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

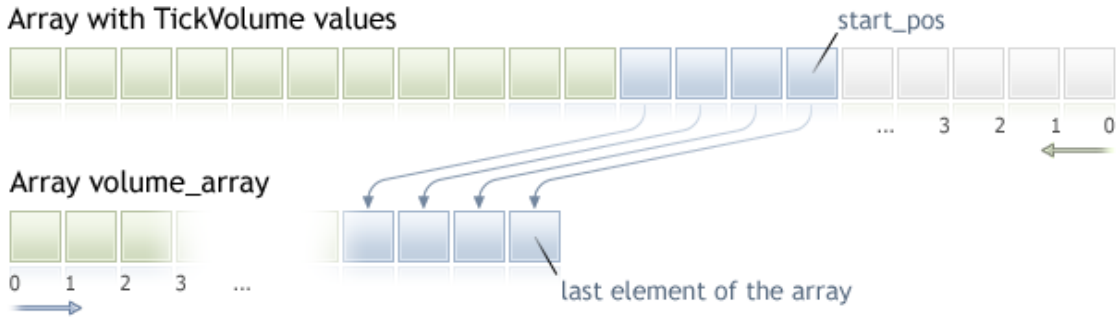
Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Vedere un esempio dettagliato di dati storici richiesti, nella sezione Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyTickVolume

La funzione ottiene in volume_array i dati storici dei volumi tick per la coppia simbolo-periodo selezionati nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyTickVolume(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   int              start_pos,        // posizione di inizio
   int              count,            // conteggio dati da copiare
   long             volume_array[]    // array target per i volumi tick
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyTickVolume(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   int              count,            // conteggio dati da copiare
   long             volume_array[]    // array target per i volumi tick
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyTickVolume(
```

```
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   datetime         stop_time,        // data ed ora di fine
   long             volume_array[]    // array target per i volumi tick
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] L'orario di avviamento per il primo elemento da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare.

volume_array[]

[out] Array odi tipo long.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Esempio:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot TickVolume
#property indicator_label1  "TickVolume"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  C'143,188,139'
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      bars=3000;
//--- buffers indicatore
double         TickVolumeBuffer[];
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

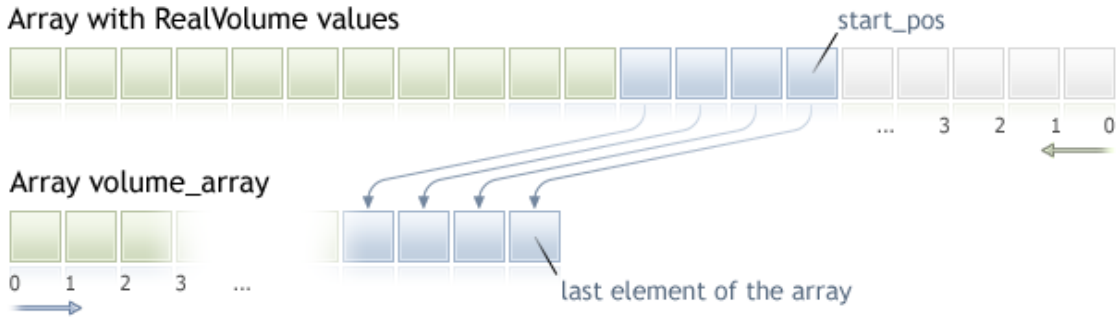
```
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      long timeseries[];
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-
      Print("We have received the following number of TickVolume values: "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedere un esempio dettagliato di dati storici richiesti, nella sezione Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyRealVolume

La funzione ottiene in volume_array i dati storici dei volumi trade per la coppia simbolo-periodo selezionati nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopyRealVolume(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   int              start_pos,        // posizione di inizio
   int              count,            // conteggio dati da copiare
   long             volume_array[]    // array target per i valuti dei volumi
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopyRealVolume(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   int              count,            // conteggio dati da copiare
   long             volume_array[]    // array target per i valuti dei volumi
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopyRealVolume(
```

```
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   datetime         stop_time,        // data ed ora di fine
   long             volume_array[]    // array target per i valuti dei volumi
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] L'orario di avviamento per il primo elemento da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare.

volume_array[]

[out] Array odi tipo long.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso dierrore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

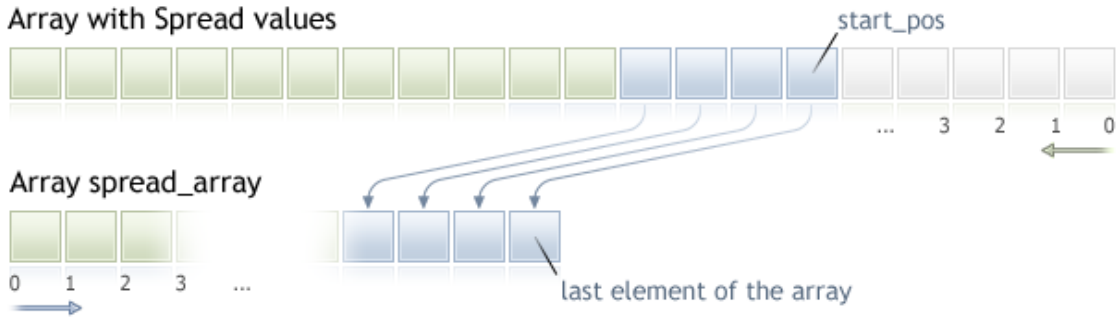
Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Vedere l'esempio di dati storici richiesti nella sezione Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopySpread

La funzione ottiene in spread_array i dati storici dei valori spread per la coppia simbolo-periodo selezionati nella quantità specificata. Occorre notare che l'ordinamento degli elementi è dal presente al passato, cioè, la posizione iniziale di 0 significa la barra corrente.



Quando si copia una quantità ancora sconosciuta di dati, si raccomanda di utilizzare gli array dinamici come array di destinazione, perché se il conteggio dei dati richiesti è di meno (o più) rispetto alla lunghezza dell'array di destinazione, la funzione cerca di riallocare la memoria in modo che i dati richiesti vi entrino completamente.

Se si conosce la quantità necessaria di dati da copiare, dovrebbe essere meglio fatto in un buffer allocato in modo statico, al fine di evitare un' allocazione eccessiva di memoria.

Non importa ciò che è di proprietà dell'array di destinazione - as_series=true o as_series=false. I dati verranno copiati in modo che l'elemento più vecchio sarà situato all'inizio della memoria fisica allocata per l'array. Ci sono 3 varianti di chiamate di funzione.

Chiamate dalla prima posizione ed il numero di elementi richiesti

```
int  CopySpread(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   int              start_pos,        // posizione di inizio
   int              count,            // conteggio dati da copiare
   int              spread_array[]    // array target per i valori spread
   );
```

Chiamata dalla data di inizio e il numero di elementi richiesti

```
int  CopySpread(
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   int              count,            // conteggio dati da copiare
   int              spread_array[]    // array target per i valori spread
   );
```

Chiamata per le date di inizio e di fine di un intervallo di tempo richiesto

```
int  CopySpread(
```

```
   string           symbol_name,      // nome del simbolo
   ENUM_TIMEFRAMES  timeframe,        // periodo
   datetime         start_time,       // data ed ora di inizio
   datetime         stop_time,        // data ed ora di fine
   int              spread_array[]    // array target per i valori spread
   );
```

Parametri

symbol_name

[in] Nome del Simbolo.

timeframe

[in] Periodo.

start_pos

[in] La posizione di partenza per il primo elemento da copiare.

count

[in] Conteggio dati da copiare.

start_time

[in] L'orario di avviamento per il primo elemento da copiare.

stop_time

[in] Orario barra, corrispondente all'ultimo elemento da copiare.

spread_array[]

[out] Array di tipo int.

Valore restituito

Restituisce il numero di dati copiati o -1 in caso di errore.

Nota

Se l'intervallo di insieme dei dati richiesti è fuori dei dati disponibili sul server, la funzione restituisce -1. Se sono richiesti i dati fuori da TERMINAL_MAXBARS (numero massimo di barre nel grafico), la funzione anche restituire -1.

Quando si richiedono dati dall'indicatore, se le timeseries richieste non sono ancora costruite o hanno bisogno di essere scaricati dal server, la funzione restituisce immediatamente -1, ma il processo di download/costruzione verrà avviato.

Quando si richiedono dati da un Expert Advisor o uno script, verrà avviato il download dal server, se il terminale non dispone di questi dati a livello locale, o la costruzione delle timeserie richieste avrà inizio, se i dati possono essere costruiti dallo storico locale, ma non sono ancora pronti. La funzione restituisce la quantità di dati che sarà pronta entro il momento del timeout dell'espirazione, ma il download dello storico continuerà, ed alla successiva richiesta simile la funzione restituisce più dati.

Quando si richiedono dati per la data di inizio ed il numero di elementi necessari, solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre minore o uguale a quello specificato.

Quando si richiedono dati in un intervallo di date specificato, solo i dati di questo intervallo verranno restituiti. L'intervallo è impostato e contato fino ai secondi. Vuol dire, che il tempo di apertura di ogni barra, per il quale viene restituito il valore (volume, spread, valore sul buffer indicatore, i prezzi Open, High, Low, Close o l'orario di apertura Time) è sempre entro l'intervallo richiesto.

Dunque, se il giorno corrente è Sabato, al tentativo di copiare i dati su un arco di tempo di settimane specificando *start_time=Last_Tuesday* e *stop_time=Last_Friday* la funzione restituirà 0, perché l'orario di apertura su timeframe di settimana è sempre Domenica, ma una barra da una settimana non cadere nell'intervallo specificato.

Se c'è bisogno di restituire il valore corrispondente alla corrente barra incompiuta, è possibile utilizzare la prima forma di chiamata specificando *start_pos*=0 e *count*= 1.

Esempio:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Spread
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      bars=3000;
//--- buffers indicatore
double         SpreadBuffer[];
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      int spread_int[];
      ArraySetAsSeries(spread_int,true);
      int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
      Print("We have received the following number of Spread values: ",spreads);
      for (int i=0;i<spreads;i++)
      {
      SpreadBuffer[rates_total-1-i]=spread_int[i];
      if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
      }
     }
   else
     {
      double Ask,Bid;
      Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
      Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
      Comment("Ask = ",Ask,"  Bid = ",Bid);
      SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedere l'esempio di dati storici richiesti nella sezione Metodi di Binding Oggetti. Lo script disponibile in questa sezione mostra come ottenere i valori degli indicatori iFractals delle ultime 1000 barre e come visualizzare gli ultimi 10 su e 10 giù frattali sul grafico. Una tecnica simile può essere utilizzata per tutti gli indicatori che hanno dati mancanti e che sono solitamente elaborati utilizzando i seguenti stili:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyTicks

La funzione riceve ticks nel formato MqlTick in ticks_array. In questo caso, i ticks sono indicizzati dal passato al presente, cioè il tick indicizzato 0 (zero) è il più vecchio nell'array. Per l'analisi del tick, controllare il campo *flags*, che mostra esattamente cosa è cambiato nel tick.

```
int  CopyTicks(
   string           symbol_name,           // Nome simbolo
   MqlTick&         ticks_array[],         // Array che riceve i Tick
   uint             flags=COPY_TICKS_ALL,  // La flag che determina il tipo di tick r
   ulong            from=0,                // La data dal quale vuoi richiedere ticks
   uint             count=0                // Il numero di ticks che vuoi ricevere
   );
```

Parametri

symbol_name

[in] Simbolo.

ticks_array

[out] Un array di tipo MqlTick per ricevere ticks.

flags

[in] Un flag per definire il tipo di ticks richiesti. COPY_TICKS_INFO - ticks con cambiamenti Bid e/o Ask , COPY_TICKS_TRADE - ticks con cambiamenti in Last e Volume, COPY_TICKS_ALL - all ticks. Per qualsiasi tipo di richiesta, i valori dei tick precedenti vengono aggiunti ai campi rimanenti della struttura MqlTick.

from

[in] La data dal quale vuoi ricchiedere ticks. In millisecondi da 1970.01.01. If *from*=0, l'ultimo *count* ticks verrà restituito.

count

[in] Il numero di ticks richiesti. Se i parametri 'from' e 'count' non sono specificati, tutti i ticks disponibili più recenti (ma non più di 2000) verranno scritti in per ticks_array [].

Valore restituito

Il numero di tick copiati o -1 in caso di errore.

Ulteriore nota

La funzione CopyTicks() permette di richiedere ed analizzare tutti i ticks ricevuti. La prima chiamata di CopyTicks() avvia la sincronizzazione del database del simbolo del tick memorizzato sul disco rigido. Se il database locale non fornisce tutti i ticks richiesti, allora i ticks mancanti verranno scaricati automaticamente dal trade server. Ticks che cominciano dalla data ***from*** specificata in CopyTicks() fino al momento corrente, verranno sincronizzati. Dopo di che, saranno aggiunti tutti i ticks in arrivo per questo simbolo al database dei ticks in modo da mantenerli nello stato sincronizzato.

Se i parametri *from* e *count* non sono stati specificati, tutti i ticks disponibili (ma non più di 2000) verranno scritti nel *ticks_array[]*. Il parametro *flags* permette di specificre il tipo di ticks richiesti.

COPY_TICKS_INFO - vengono restituiti ticks con cambiamento di prezzo Bid e/o Ask . Saranno inoltre aggiunti i dati di altri campi. Ad esempio, se solo Bid è cambiato, i campi *ask* e *volume* saranno riempiti con ultimi valori noti. Per sapere esattamente che cosa è cambiato, analizzare il campo *flag*, che avrà il valore di TICK_FLAG_BID e/o TICK_FLAG_ASK. Se un tick ha valore zero dei prezzi Bid ed Ask, e le flags mostrano che questi dati sono cambiati (*flags*= TICK_FLAG_BID | TICK_FLAG_ASK), ciò significa che il libro degli ordini (Market Depth - Profondità Mercato) è vuoto. In altre parole, non ci sono ordini d' acquisto(buy) e vendite(sell).

COPY_TICKS_TRADE - vengono restituiti ticks con i cambiamenti di prezzo Last e Volume. Saranno inoltre aggiunti i dati di altri campi, vale a dire gli ultimi noti valori di Bid ed Ask che saranno specificati nei campi appropriati. Per sapere esattamente che cosa è cambiato, analizzare il campo *field*, che avrà il valore TICK_FLAG_LAST e TICK_FLAG_VOLUME.

COPY_TICKS_ALL - vengono restituiti tutti i ticks con qualsiasi cambiamento. I cambi non cambiati verranno riempiti con gli ultimi valori noti.

La chiamata di CopyTicks() con il flag COPY_TICKS_ALL ritorna immediatamente tutti i ticks dall' intervallo della richiesta, mentre le chiamate in altre modalità richiedono un certo tempo per elaborare e selezionare i ticks, quindi non forniscono un significativo vantaggio di velocità.

Quando si richiedono i ticks (sia con COPY_TICKS_INFO che COPY_TICKS_TRADE), ogni ticke conterrà le piene informazioni di prezzo così come l'orario del tick (*bid, ask, last* e *volume*). Questa funzione viene fornita per facilitare l'analisi dello stato di trade al momento di ogni tick, quindi non è necessario richiedere una storia tick profonda e cercare i valori di altri campi.

Negli indicatori, la funzione CopyTicks() restituisce il risultato: quando viene chiamata da un indicatore, CopyTick() restituisce immediatamente tutti i tick disponibili di un simbolo, e lancerà la sincronizzazione del database ticks, se i dati disponibili non sono sufficienti. Tutti gli indicatori in un simbolo operano in un thread comune, in modo che l'indicatore non attenda il completamento della sincronizzazione. Dopo la sincronizzazione, CopyTicks() restituirà tutti i ticks richiesti durante la chiamata successiva. Negli indicatori, la funzione OnCalculate() è chiamata dopo l'arrivo di ogni tick.

CopyTicks() può attendere il risultato per 45 secondi in Expert Advisors e scripts: a differenza degli indicatori, ogni Expert Advisor e script operano in un thread separato, e quindi possono aspettare 45 secondi fino al completamento della sincronizzazione. Se la quantità necessaria di ticks non riesce ad essere sincronizzata durante questo tempo, CopyTicks() restituirà i ticks disponibili per timeout e continuerà la sincronizzazione. OnTick() in Expert Advisor non è un handler per ogni tick, essa solo notifica ad un Expert Advisor i cambiamenti nel mercato. Può essere una serie di cambiamenti: il terminale può contemporaneamente creare alcuni ticks, ma onTick() sarà chiamato solo una volta a notificare all' EA dello stato attuale del mercato.

Il tasso di restituzione dei dati: il terminale conserva nella cache l'accesso veloce a 4096 ultimi ticks per ogni strumento (65536 ticks per simboli con corrente Market Depth). Se i ticks richiesti per la sessione di trading corrente sono oltre la cache, CopyTicks() chiama i ticks memorizzati nella memoria del terminale. Queste richieste richiedono più tempo per l'esecuzione. Le richieste più lente sono quelle che richiedono i ticks per altri giorni, dal momento che i dati vengono letti dal disco in questo caso.

Esempio:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property script_show_inputs
//--- Requesting 100 million ticks to be sure we receive the entire tick history
input int      getticks=100000000; // The number of required ticks
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int     attempts=0;     // Count of attempts
   bool    success=false;  // The flag of a successful copying of ticks
   MqlTick tick_array[];   // Tick receiving array
   MqlTick lasttick;       // To receive last tick data
   SymbolInfoTick(_Symbol,lasttick);
//--- Make 3 attempts to receive ticks
   while(attempts<3)
     {
      //--- Measuring start time before receiving the ticks
      uint start=GetTickCount();
//--- Requesting the tick history since 1970.01.01 00:00.001 (parameter from=1 ms)
      int received=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,1,getticks);
      if(received!=-1)
        {
         //--- Showing information about the number of ticks and spent time
         PrintFormat("%s: received %d ticks in %d ms",_Symbol,received,GetTickCount()-
         //--- If the tick history is synchronized, the error code is equal to zero
         if(GetLastError()==0)
           {
            success=true;
            break;
           }
         else
            PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d
            _Symbol,received,GetTickCount()-start,_LastError);
        }
      //--- Counting attempts
      attempts++;
      //--- A one-second pause to wait for the end of synchronization of the tick data
      Sleep(1000);
     }
//--- Receiving the requested ticks from the beginning of the tick history failed in t
   if(!success)
     {
      PrintFormat("Error! Failed to receive %d ticks of %s in three attempts",getticks
      return;
     }
   int ticks=ArraySize(tick_array);
//--- Showing the time of the first tick in the array
   datetime firstticktime=tick_array[ticks-1].time;
```

```
   PrintFormat("Last tick time = %s.%03I64u",
               TimeToString(firstticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar
//--- выведем время последнего тика в массиве
   datetime lastticktime=tick_array[0].time;
   PrintFormat("First tick time = %s.%03I64u",
               TimeToString(lastticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar

//---
   MqlDateTime today;
   datetime current_time=TimeCurrent();
   TimeToStruct(current_time,today);
   PrintFormat("current_time=%s",TimeToString(current_time));
   today.hour=0;
   today.min=0;
   today.sec=0;
   datetime startday=StructToTime(today);
   datetime endday=startday+24*60*60;
   if((ticks=CopyTicksRange(_Symbol,tick_array,COPY_TICKS_ALL,startday*1000,endday*100
     {
      PrintFormat("CopyTicksRange(%s,tick_array,COPY_TICKS_ALL,%s,%s) failed, error %d
                  _Symbol,TimeToString(startday),TimeToString(endday),GetLastError());
      return;
     }
   ticks=MathMax(100,ticks);
//--- Showing the first 100 ticks of the last day
   int counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         counter++;
         PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
        }
     }
//--- Showing the first 100 deals of the last day
   counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         if(((tick_array[i].flags&TICK_FLAG_BUY)==TICK_FLAG_BUY) || ((tick_array[i].fl
           {
            counter++;
            PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
           }
        }
     }
```

```
  }
//+------------------------------------------------------------------+
//| Returns the string description of a tick                         |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03d ",
                            TimeToString(tick.time),tick.time_msc%1000);
//--- Checking flags
   bool buy_tick=((tick.flags&TICK_FLAG_BUY)==TICK_FLAG_BUY);
   bool sell_tick=((tick.flags&TICK_FLAG_SELL)==TICK_FLAG_SELL);
   bool ask_tick=((tick.flags&TICK_FLAG_ASK)==TICK_FLAG_ASK);
   bool bid_tick=((tick.flags&TICK_FLAG_BID)==TICK_FLAG_BID);
   bool last_tick=((tick.flags&TICK_FLAG_LAST)==TICK_FLAG_LAST);
   bool volume_tick=((tick.flags&TICK_FLAG_VOLUME)==TICK_FLAG_VOLUME);
//--- Checking trading flags in a tick first
   if(buy_tick || sell_tick)
     {
      //--- Forming an output for the trading tick
      desc=desc+(buy_tick?StringFormat("Buy Tick: Last=%G Volume=%d ",tick.last,tick.v
      desc=desc+(sell_tick?StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,tick
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+"(Trade tick)";
     }
   else
     {
      //--- Form a different output for an info tick
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+(last_tick?StringFormat("Last=%G ",tick.last):"");
      desc=desc+(volume_tick?StringFormat("Volume=%d ",tick.volume):"");
      desc=desc+"(Info tick)";
     }
//--- Returning tick description
   return desc;
  }
//+------------------------------------------------------------------+
/* Example of the output
Si-12.16: received 11048387 ticks in 4937 ms
Last tick time = 2016.09.26 18:32:59.775
First tick time = 2015.06.18 09:45:01.000
1.  2016.09.26 09:45.249 Ask=65370 Bid=65370 (Info tick)
2.  2016.09.26 09:47.420 Ask=65370 Bid=65370 (Info tick)
3.  2016.09.26 09:50.893 Ask=65370 Bid=65370 (Info tick)
4.  2016.09.26 09:51.827 Ask=65370 Bid=65370 (Info tick)
5.  2016.09.26 09:53.810 Ask=65370 Bid=65370 (Info tick)
6.  2016.09.26 09:54.491 Ask=65370 Bid=65370 (Info tick)
7.  2016.09.26 09:55.913 Ask=65370 Bid=65370 (Info tick)
```

```
8.  2016.09.26 09:59.350 Ask=65370 Bid=65370 (Info tick)
9.  2016.09.26 09:59.678 Bid=65370 (Info tick)
10. 2016.09.26 10:00.000 Sell Tick: Last=65367 Volume=3 (Trade tick)
11. 2016.09.26 10:00.000 Sell Tick: Last=65335 Volume=45 (Trade tick)
12. 2016.09.26 10:00.000 Sell Tick: Last=65334 Volume=95 (Trade tick)
13. 2016.09.26 10:00.191 Sell Tick: Last=65319 Volume=1 (Trade tick)
14. 2016.09.26 10:00.191 Sell Tick: Last=65317 Volume=1 (Trade tick)
15. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=1 (Trade tick)
16. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=10 (Trade tick)
17. 2016.09.26 10:00.191 Sell Tick: Last=65315 Volume=5 (Trade tick)
18. 2016.09.26 10:00.191 Sell Tick: Last=65313 Volume=3 (Trade tick)
19. 2016.09.26 10:00.191 Sell Tick: Last=65307 Volume=25 (Trade tick)
20. 2016.09.26 10:00.191 Sell Tick: Last=65304 Volume=1 (Trade tick)
21. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=1 (Trade tick)
22. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=10 (Trade tick)
23. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=5 (Trade tick)
24. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=1 (Trade tick)
25. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=6 (Trade tick)
26. 2016.09.26 10:00.191 Sell Tick: Last=65299 Volume=1 (Trade tick)
27. 2016.09.26 10:00.191 Bid=65370 (Info tick)
28. 2016.09.26 10:00.232 Ask=65297 (Info tick)
29. 2016.09.26 10:00.276 Sell Tick: Last=65291 Volume=31 (Trade tick)
30. 2016.09.26 10:00.276 Sell Tick: Last=65290 Volume=1 (Trade tick)
*/
```

Guarda anche

SymbolInfoTick, Struttura per i Prezzi Correnti, OnTick()

CopyTicksRange

La funzione riceve ticks in formato MqlTick nell'intervallo di date specificato in ticks_array. Indicizzazione va dal passato al presente che significa che un tick con l'indice 0 è il più vecchio nell'array. Per l'analisi dei tick, controllare la il campo *flag*, che mostra cosa esattamente è cambiato.

```
int  CopyTicksRange(
   const string      symbol_name,           // nome del simbolo
   MqlTick&          ticks_array[],         // array ricevente i tick
   uint              flags=COPY_TICKS_ALL,  // flag che definisce il tipo di ticks che
   ulong             from_msc=0,            // data, dalla quale i ticks vengono richie
   ulong             to_msc=0               // data, fino a quando i ticks vengono rich
   );
```

parametri

symbol_name

[in] Simbolo.

ticks_array

[out] MqlTick array statico o dinamicoper il ricevimento dei ticks. Se l'array statico non può contenere tutti i ticks dall'intervallo di tempo richiesto, viene ricevuto l'ammontare massimo possibile di ticks. In questo caso, la funzione genera l'errore ERR_HISTORY_SMALL_BUFFER (4407).

flags

[in] Una flag per definire il tipo di ticks richiesti. COPY_TICKS_INFO - ticks con cambiamenti Bid e/o Ask, COPY_TICKS_TRADE - ticks con i cambiamenti in Last e Volume, COPY_TICKS_ALL - tutti i ticks. Per qualsiasi tipo di richiesta, si aggiungono i valori del precedente tick ai campi rimanenti della struttura MqlTick.

from_msc

[in] La data, dalla quale si desidera richiedere i ticks. In millisecondi dal 1970/01/01. Se il parametro *from_msc* non è specificato, vengono inviati i ticks dall'inizio dello storico. Ticks con l'orario >= from_msc vengono inviati.

to_msc

[in] La data, fino al quale si desidera richiedere ticks. In millisecondi dal 01.01.1970. Vengono inviati Ticks con l'orario <= to_msc (ad_msc). Se il paramtro *to_msc* non è specificato, vengono inviati tutti i ticks fino alla fine dello storico.

Valore di ritorno

Il numero di tick copiato oppure -1 in caso di errore. GetLastError() è in grado di restituire i seguenti errori:

- ERR_HISTORY_TIMEOUT - il tempo di attesa della sincronizzazione ticks è scaduto, la funzione ha inviato tutto quello che aveva.
- ERR_HISTORY_SMALL_BUFFER - il buffer statico è troppo piccolo. Solo l'ammontare che l'array piò memorizzare è stato inviato.
- ERR_NOT_ENOUGH_MEMORY - memoria insufficiente per la ricezione dello storico dal range specificato all' array dinamico dei tick. Impossibile allocare la memoria sufficiente per l'array tick.

Nota

La funzione CopyTicksRange() viene utilizzata per richiedere ticks strettamente da un intervallo specifico, per esempio, da un certo giorno nello storico. Allo stesso tempo, CopyTicks() permette di specificare solo la data di inizio, per esempio - ricevere tutti i ticks dall'inizio del mese fino al momento attuale.

Guarda anche

SymbolInfoTick, Struttura per i Prezzi Correnti, OnTick, CopyTicks

iBars

Restituisce il numero di barre di un simbolo e di un periodo corrispondenti, disponibili nello storico.

```
int  iBars(
   const string           symbol,          // Simbolo
   ENUM_TIMEFRAMES        timeframe        // Periodo
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

Valore di ritorno

Il numero di barre di un simbolo e di un periodo corrispondenti, disponibili nello storico, ma non più di quanto consentito dal parametro "Max barre nel chart" nelle impostazioni della piattaforma.

Esempio:

```
  Print("Il conteggio Barre su 'EURUSD,H1' è ",iBars("EURUSD",PERIOD_H1));
```

Guarda anche

Bars

iBarShift

Cerca la barra per orario. La funzione restituisce l'indice della barra corrispondente al tempo/orario specificato.

```
int  iBarShift(
   const string        symbol,          // Simbolo
   ENUM_TIMEFRAMES     timeframe,       // Periodo
   datetime            time,            // Orario
   bool                exact=false      // Modalità
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. PERIOD_CURRENT significa il periodo chart corrente.

time

[in] Valore temporale da cercare.

exact=false

[in] Un valore di ritorno, nel caso in cui la barra con l'ora specificata non venga trovata. Se *exact=false*, *iBarShift* restituisce l'indice della barra più vicina, l'orario Open (di apertura) è inferiore all'orario specificato (*time_open<time*). Se tale barra non viene trovata (lo storico prima dell'orario specificato non è disponibile), la funzione restituisce -1. Se *exact=true*, iBarShift non cerca la barra più vicina, ma restituisce immediatamente -1.

Valore di ritorno

L'indice della barra corrispondente all'orario specificato. Se la barra corrispondente all'orario specificato non viene trovata (c'è una lacuna nello storico), la funzione restituisce -1 o l'indice della barra più vicina (a seconda del parametro *'exact'*).

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- La data è domenica
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- Se non c'è alcuna barra all'ora specificata, iBarShift restituirà l'indice della
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- Controlla il codice di errore dopo la chiamata di iBarShift()
```

```
   int error=GetLastError();
   if(error!=0)
     {
      PrintFormat("iBarShift(): GetLastError=%d - La data richiesta %s "+
                  "per %s %s non è stata trovata nello storico disponibile",
                  error,TimeToString(time),symbol,EnumToString(tf));
      return;
     }
//--- La funzione iBarShift() è stata eseguita correttamente, restituisce un risultato
   PrintFormat("1. %s %s %s(%s): l'indice della barra è %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),
               DayOfWeek(time),bar_index,string(exact));
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("L'orario della barra #%d è %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//--- Richiede l'indice della barra con l'orario specificato; se non ci sono barre, ve
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- La funzione iBarShift() è stata eseguita correttamente, restituisce un risultato
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//| Restituisce il nome del giorno della settimana                   |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
```

```
   return day;
  }
//+------------------------------------------------------------------+
/* Risultato dell'esecuzione
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): l'indice della barra è 64 (exact=fals
   Orario della barra #64 è 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):l'indice della barra è -1 (exact=true
*/
```

iClose

Restituisce il prezzo Close (di chiusura) della barra (indicato dal parametro 'shift') sul chart corrispondente.

```
double  iClose(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il prezzo Close (di chiusura) della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
```

```
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopyClose, CopyRates

iHigh

Restituisce il prezzo High della barra (indicata dal parametro 'shift') sul chart corrispondente.

```
double  iHigh(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il prezzo High della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopyHigh, CopyRates

iHighest

Restituisce l'indice del valore più alto trovato sul chart corrispondente (spostamento relativo alla barra corrente).

```
int  iHighest(
   const string        symbol,              // Simbolo
   ENUM_TIMEFRAMES     timeframe,           // Periodo
   ENUM_SERIESMODE     type,                // Identificatore Timeseries
   int                 count=WHOLE_ARRAY,   // Numero di elementi
   int                 start=0              // Indice
  );
```

Parametri

symbol

[in] Il simbolo, su cui verrà eseguita la ricerca. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

type

[in] L'identificatore delle timeseries, in cui verrà eseguita la ricerca. Può essere uguale a qualsiasi valore da ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Il numero di elementi nelle timeseries (dalla barra corrente verso la direzione crescente dell'indice), tra i quali deve essere eseguita la ricerca.

start=0

[in] L'indice (slittamento relativo alla barra corrente) della barra iniziale, da cui inizia la ricerca del valore più alto. I valori negativi vengono ignorati e sostituiti con un valore zero.

Valore di ritorno

L'indice del valore più alto trovato sul chart corrispondente (slittamento relativo alla barra corrente) o -1 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Esempio:

```
   double val;
//--- Calcolo del valore di chiusura Close più alto tra 20 barre consecutive
//--- Dall'indice 4 all'indice 23 incluso, nel timeframe corrente
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("iHighest() - errore di chiamata. Codice errore=%d",GetLastError());
```

iLow

Restituisce il prezzo Low della barra (indicato dal parametro 'shift') sul chart corrispondente.

```
double  iLow(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il prezzo Low della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopyLow, CopyRates

iLowest

Restituisce l'indice del valore più piccolo trovato sul chart corrispondente (slittamento relativo alla barra corrente).

```
int  iLowest(
   const string        symbol,              // Simbolo
   ENUM_TIMEFRAMES     timeframe,           // Periodo
   ENUM_SERIESMODE     type,                // Identificatore Timeseries
   int                 count=WHOLE_ARRAY,   // Numero di elementi
   int                 start=0              // Indice
  );
```

Parametri

symbol

[in] Il simbolo, su cui verrà eseguita la ricerca. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 indica il periodo del chart corrente.

type

[in] L'identificatore delle timeseries, in cui verrà eseguita la ricerca. Può essere uguale a qualsiasi valore da ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Il numero di elementi nelle timeseries (dalla barra corrente verso la direzione crescente dell'indice), tra i quali deve essere eseguita la ricerca.

start=0

[in] L'indice (slittamento relativo alla barra corrente) della barra iniziale, da cui inizia la ricerca del valore più basso. I valori negativi vengono ignorati e sostituiti con un valore zero.

Valore di ritorno

L'indice del valore più basso trovato sul chart corrispondente (slittamento relativo alla barra corrente) o -1 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Esempio:

```
   double val;
//--- Cerca una barra con il valore più basso del volume reale tra 15 barre consecutiv
//--- Dall'indice 10 all'indice 24 incluso, nel timeframe corrente
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("iLowest() - errore di chiamata. Codice errore=%d",GetLastError());
```

iOpen

Restituisce il prezzo Open della barra (indicata dal parametro 'shift') sul chart corrispondente.

```
double  iOpen(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il prezzo Open della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopyOpen, CopyRates

iTime

Restituisce il tempo di apertura della barra (indicato dal parametro 'shift') sul chart corrispondente.

```
datetime  iTime(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il tempo di apertura della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione Start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- La data è domenica
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- non c'è una barra all'orario specificato, iBarShift restituirà l'indice della ba
   int bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("1. %s %s %s(%s): l'indice della barra è %d (exact=%s)",
```

```
                symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("L'orario della barra #%d è %s (%s)",
                bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//PrintFormat(iTime(symbol,tf,bar_index));
//--- Richiede l'indice della barra con il timeframe specificato; ma non c'è la barra,
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
                symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
  }
//+------------------------------------------------------------------+
//| Restituisce il nome del giorno della settimana                   |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
   return day;
  }
/* Il risultato:
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

Guarda anche

CopyTime, CopyRates

iTickVolume

Restituisce il volume tick della barra (indicato dal parametro 'shift') sul chart corrispondente.

```
long  iTickVolume(
   const string        symbol,          // Simbolo
   ENUM_TIMEFRAMES     timeframe,       // Periodo
   int                 shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il volume del tick della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopyTickVolume, CopyRates

iRealVolume

Restituisce il volume reale della barra (indicato dal parametro 'shift') sul chart corrispondente.

```
long  iRealVolume(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il volume reale della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopyRealVolume, CopyRates

iVolume

Restituisce il volume tick della barra (indicato dal parametro 'shift') sul chart corrispondente.

```
long  iVolume(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il volume del tick della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopyTickVolume, CopyRates

iSpread

Restituisce il valore di spread della barra (indicato dal parametro 'shift') sul chart corrispondente.

```
long  iSpread(
   const string         symbol,          // Simbolo
   ENUM_TIMEFRAMES      timeframe,       // Periodo
   int                  shift            // Slittamento
   );
```

Parametri

symbol

[in] Il nome simbolico dello strumento finanziario. NULL significa il simbolo corrente.

timeframe

[in] Periodo. Può essere uno dei valori dell'enumerazione ENUM_TIMEFRAMES. 0 significa il periodo del chart corrente.

shift

[in] L'indice del valore ricevuto dalla timeseries (spostamento all'indietro di un numero specificato di barre rispetto alla barra corrente).

Valore di ritorno

Il valore Spread della barra (indicato dal parametro 'shift') sul chart corrispondente o 0 in caso di errore. Per i dettagli dell'errore, chiamare la funzione GetLastError().

Nota

La funzione restituisce sempre i dati effettivi. A tale scopo esegue una richiesta alle timeseries per il simbolo/periodo specificato durante ogni chiamata. Ciò significa che se non ci sono dati pronti durante la prima chiamata di funzione, potrebbe essere necessario un po' di tempo per preparare il risultato.

La funzione non memorizza i risultati delle chiamate precedenti e non esiste una cache locale per il ritorno rapido dei valori.

Esempio:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Funzione-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "   ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "   ,DoubleToString(open,Digits()),"\n",
           "High: "   ,DoubleToString(high,Digits()),"\n",
           "Low: "    ,DoubleToString(low,Digits()),"\n",
           "Close: "  ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "   ,IntegerToString(bars),"\n"
           );
  }
```

Guarda anche

CopySpread, CopyRates

Simboli Personalizzati

Funzioni per la creazione e la modifica delle proprietà dei simboli personalizzati.

Quando si collega il terminale a un determinato trade server, un utente è in grado di lavorare con le time series dei simboli finanziari forniti da un broker. I simboli finanziari disponibili vengono visualizzati come un elenco nella finestra del Market Watch. Un gruppo separato di funzioni consente di ricevere dati sulle proprietà dei simboli, sessioni di trading ed aggiornamenti della profondità di mercato(market depth).

Il gruppo di funzioni descritte in questa sezione consente la creazione di simboli personalizzati. A tal fine, gli utenti possono applicare i simboli esistenti del trade server, i file di testo o le origini dati esterne.

Funzione	Azione
CustomSymbolCreate	Creare un simbolo personalizzato con il nome specificato nel gruppo specificato
CustomSymbolDelete	Eliminare un simbolo personalizzato con il nome specificato
CustomSymbolSetInteger	Impostare il valore di proprietà del tipo intero per un simbolo personalizzato
CustomSymbolSetDouble	Impostare il valore di proprietà del tipo reale per un simbolo personalizzato
CustomSymbolSetString	Impostare il valore della proprietà di tipo stringa per un simbolo personalizzato
CustomSymbolSetMarginRate	Impostare i tassi di margine in base al tipo di ordine e alla direzione di un simbolo personalizzato
CustomSymbolSetSessionQuote	Imposta l'ora di inizio e fine della sessione di quotazioni specificata per il simbolo ed il giorno della settimana specificati
CustomSymbolSetSessionTrade	Impostare l'ora di inizio e fine della sessione di trading specificata per il simbolo e il giorno della settimana specificati
CustomRatesDelete	Elimina tutte le barre dalla cronologia dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato
CustomRatesReplace	Sostituisce completamente lo storico dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato con i dati dell'array tipo MqlRates
CustomRatesUpdate	Aggiunge barre mancanti allo storico dei simboli custom e sostituisce i dati esistenti con quelli dell'array tipo MqlRates
CustomTicksAdd	Aggiunge i dati da un array del tipo MqlTick alla cronologia dei prezzi di un simbolo personalizzato. Il simbolo personalizzato deve essere selezionato nella finestra del Watch Market.
CustomTicksDelete	Eliminare tutti i ticks dallo storico dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato

Funzione	Azione
CustomTicksReplace	Sostituisce completamente lo storico dei prezzi del simbolo personalizzato entro l'intervallo di tempo specificato con i dati dell'array di tipo MqlTick
CustomBookAdd	Passa lo status del Depth of Market per un simbolo personalizzato

CustomSymbolCreate

Crea un simbolo personalizzato con il nome specificato nel gruppo specificato.

```
bool  CustomSymbolCreate(
   const string      symbol_name,          // nome del simbolo personalizzato
   const string      symbol_path="",       // nome di un gruppo in cui dev essere creato
   const string      symbol_origin=NULL    // nome del simbolo usato come base per crea
   );
```

Parametri

symbol_name

[in] Nome simbolo personalizzato. Non dovrebbe contenere gruppi o sottogruppi in cui si trova il simbolo.

symbol_path=""

[in] Il nome del gruppo in cui si trova un simbolo.

symbol_origin=NULL

[in] Nome del simbolo le cui proprietà di un simbolo personalizzato creato devono essere copiate. Dopo aver creato un simbolo personalizzato, qualsiasi valore di proprietà può essere modificato in uno necessario utilizzando le funzioni appropriate.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Tutti i simboli personalizzati vengono creati nella sezione speciale personalizzata. Se il nome di un gruppo non è specificato (il parametro *symbol_path* nella funzione CustomSymbolCreate contiene una stringa vuota o NULL), un simbolo personalizzato viene generato nella sezione principale Personalizzata. Qui possiamo tracciare un'analogia con il file system, dove gruppi e sottogruppi possono essere visualizzati come cartelle e sottocartelle

I nomi di simboli e gruppi possono contenere solo lettere latine senza punteggiatura, spazi o caratteri speciali (possono contenere solo ".", "_", "&" E "#"). Non è raccomandato l'uso dei caratteri <, >, :, ", /, |, ?, *.

Il nome del simbolo personalizzato deve essere univoco indipendentemente dal nome del gruppo in cui è stato creato. Se esiste già un simbolo con lo stesso nome, la funzione CustomSymbolCreate() restituisce "false", mentre la successiva chiamata GetLastError() restituisce l'errore 5300 (ERR_NOT_CUSTOM_SYMBOL) o 5304 (ERR_CUSTOM_SYMBOL_EXIST).

La lunghezza del nome del simbolo non deve superare 31 caratteri. In caso contrario, CustomSymbolCreate() restituisce 'false' e l'errore 5302 - ERR_CUSTOM_SYMBOL_NAME_LONG viene attivato.

Il parametro *symbol_path* può essere impostato in due modi:

- solo un nome di gruppo senza un nome del simbolo personalizzato, ad esempio - "CFD\\Metals". È meglio usare questa opzione per evitare errori.

- o nome <gruppo> + separatore gruppi "\\" +<nome simbolo personalizzato> ad esempio - "CFD\\Metals\\Platinum". In questo caso, il nome del gruppo dovrebbe terminare con il nome esatto del simbolo personalizzato. In caso di mancata corrispondenza, il simbolo personalizzato viene comunque creato, ma non nel gruppo desiderato. Ad esempio, se *symbol_path*="CFD\\Metals**P**latinum" e *symbol_name*="**p**latinum" (register error), allora viene creato un simbolo personalizzato denominato "platinum" nel gruppo "Custom\CFD\Metals\Platinum". La funzione SymbolInfoGetString("platinum", SYMBOL_PATH) restituisce il valore "Custom\CFD\Metals\Platinum\platinum".

Si noti che la proprietà SYMBOL_PATHrestituisce il percorso con il nome del simbolo alla fine. Pertanto, non può essere copiato senza modifiche se si desidera creare un simbolo personalizzato nello stesso identico gruppo. In questo caso, è necessario tagliare il nome del simbolo per non ottenere il risultato sopra descritto.

Se un simbolo inesistente è impostato come *symbol_origin* parametro, quindi il simbolo personalizzato viene creato vuoto come se il *symbol_origin* parametro non è impostato. In questo caso viene attivato l'errore 4301 - ERR_MARKET_UNKNOWN_SYMBOL.

La lunghezza del parametro *symbol_path* non deve superare i 127 caratteri considerando "Custom\\", "\\" i separatori dei gruppi e il nome del simbolo se è specificato alla fine.

Guarda anche

SymbolName, SymbolSelect, CustomSymbolDelete

CustomSymbolDelete

Elimina un simbolo personalizzato con il nome specificato.

```
bool  CustomSymbolDelete(
   const string      symbol_name          // nome del simobolo personalizzato
   );
```

Parametri

symbol

[in] Nome simbolo personalizzato. Non deve corrispondere al nome di un simbolo già esistente.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Non è possibile eliminare il simbolo personalizzato visualizzato nel Market Watch o quello in cui è aperto un chart.

Guarda anche

SymbolName, SymbolSelect, CustomSymbolCreate

CustomSymbolSetInteger

Imposta il valore della proprietà di tipo integer per un simbolo personalizzato.

```
bool  CustomSymbolSetInteger(
   const string              symbol_name,      // nome del simbolo
   ENUM_SYMBOL_INFO_INTEGER  property_id,      // ID proprietà
   long                      property_value    // valore proprietà
   );
```

Parametri

symbol_name

[in] Nome simbolo personalizzato.

property_id

[in] ID della proprietà del Simbolo. Il valore può essere uno dei valori dell'enumerazione ENUM_SYMBOL_INFO_INTEGER.

property_value

[in] Una variabile di tipo long contenente il valore della proprietà.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

La cronologia(storico) dei minuti e dei tick del simbolo personalizzato viene completamente rimossa se una qualsiasi di queste proprietà viene modificata nelle specifiche del simbolo:

- SYMBOL_CHART_MODE - tipo di prezzo per la costruzione di barre (Bid o Last)
- SYMBOL_DIGITS - numero di cifre dopo il punto decimale per visualizzare il prezzo

Dopo aver eliminato la cronologia(storico) dei simboli personalizzati, il terminale tenta di creare una nuova cronologia(storico) utilizzando le proprietà aggiornate. Lo stesso accade quando le proprietà dei simboli personalizzati vengono modificate manualmente.

Guarda anche

SymbolInfoInteger

CustomSymbolSetDouble

Imposta il valore della proprietà di tipo real per un simbolo personalizzato.

```
bool  CustomSymbolSetDouble(
   const string              symbol_name,      // nome del simbolo
   ENUM_SYMBOL_INFO_DOUBLE   property_id,      // ID proprietà
   double                    property_value    // valore proprietà
   );
```

Parametri

symbol_name

[in] Nome simbolo personalizzato.

property_id

[in] ID della proprietà del Simbolo. Il valore può essere uno dei valori dell'enumerazione ENUM_SYMBOL_INFO_DOUBLE.

property_value

[in] Una variabile di tipo double contenente il valore della proprietà.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

La cronologia(storico) dei minuti e dei tick del simbolo personalizzato viene completamente rimossa se una qualsiasi di queste proprietà viene modificata nelle specifiche del simbolo:

- SYMBOL_POINT - valore un punto
- SYMBOL_TRADE_TICK_SIZE - valore di un tick che specifica la variazione di prezzo minima consentita
- SYMBOL_TRADE_TICK_VALUE - valore di variazione del prezzo di un-tick, per una posizione redditizia

Dopo aver eliminato la cronologia(storico) dei simboli personalizzati, il terminale tenta di creare una nuova cronologia(storico) utilizzando le proprietà aggiornate. Lo stesso accade quando le proprietà dei simboli personalizzati vengono modificate manualmente.

Guarda anche

SymbolInfoDouble

CustomSymbolSetString

Imposta il valore della proprietà tipo string per un simbolo personalizzato.

```
bool  CustomSymbolSetString(
   const string              symbol_name,      // nome del simbolo
   ENUM_SYMBOL_INFO_STRING   property_id,      // ID proprietà
   string                    property_value    // valore proprietà
   );
```

Parametri

symbol_name

[in] Nome simbolo personalizzato.

property_id

[in] ID della proprietà del Simbolo. Il valore può essere uno dei valori dell'enumerazione ENUM_SYMBOL_INFO_STRING.

property_value

[in] Una variabile di tipo string contenente il valore della proprietà.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

La cronologia dei minuti e dei segni di graduazione del simbolo personalizzato viene completamente rimossa se la proprietà SYMBOL_FORMULA (che imposta l'equazione per la costruzione del prezzo del simbolo personalizzato) viene modificata nella specifica del simbolo. Dopo aver eliminato lo storico(cronologia) dei simboli personalizzati, il terminale tenta di creare un nuovo storico utilizzando la nuova equazione. Lo stesso accade quando l'equazione del simbolo personalizzato viene modificata manualmente.

Guarda anche

SymbolInfoString

CustomSymbolSetMarginRate

Imposta i tassi di margine in base al tipo di ordine ed alla direzione di un simbolo personalizzato.

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // nome del simbolo
   ENUM_ORDER_TYPE    order_type,               // tipo di ordine
   double             initial_margin_rate,      // tasso del margine iniziale
   double             maintenance_margin_rate   // tasso del margine di mantenimento
   );
```

Parametri

symbol_name

[in] Nome simbolo personalizzato.

order_type

[in] Tipo di Ordine.

initial_margin_rate

[in] Una variabile di tipo double con una tasso di margine iniziale. Il margine iniziale è un deposito cauzionale per un affare di 1 lotto nella direzione appropriata. Moltiplicando il tasso dal margine iniziale, riceviamo l'importo dei fondi da riservare all'account quando piazziamo un ordine del tipo specificato.

maintenance_margin_rate

[in] Una variabile di tipo double con una tasso di margine di mantenimento. Il margine di mantenimento è un importo minimo per mantenere una posizione aperta di 1 lotto nella direzione appropriata. Moltiplicando il tasso dal margine di mantenimento, riceviamo l'importo dei fondi da riservare all'account dopo l'attivazione di un ordine del tipo specificato.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Guarda anche

SymbolInfoMarginRate

CustomSymbolSetSessionQuote

Imposta l'ora di inizio e fine della sessione di quotazione specificata per il simbolo specificato e il giorno della settimana.

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // nome del simbolo
   ENUM_DAY_OF_WEEK  day_of_week,          // giorno della settimana
   uint              session_index,        // indice della sessione
   datetime          from,                 // orario d'inizio della sessione
   datetime          to                    // orario di fine della sessione
   );
```

Parametri

symbol_name

[in] Nome simbolo personalizzato.

ENUM_DAY_OF_WEEK

[in] Giorno della settimana, valore dall'enumerazione ENUM_DAY_OF_WEEK.

uint

[in] Indice della sessione, per il quale devono essere impostate le ore di inizio e fine. L'indicizzazione della sessione inizia da 0.

from

[in] Orario di inizio sessione in secondi da 00:00, il valore dati della variabile viene ignorato.

to

[in] Orario di fine sessione in secondi da 00:00, il valore dei dati nella variabile viene ignorato.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Se la sessione con lo specificato *session_index* già esiste, la funzione modifica semplicemente l'inizio e la fine della sessione.

Se sono stati passati i parametri di partenza e di fine, zero, per la sessione, (*from=0* e *to=0*), la sessione appropriata con il *session_index* viene eliminata, mentre l'indicizzazione della sessione viene spostata verso il basso.

Le sessioni possono essere aggiunte solo sequenzialmente. In altre parole, puoi aggiungere *session_index=1* solo se la sessione con l'indice 0 esiste già. Se questa regola è interrotta, non viene creata una nuova sessione, e la funzione restituisce "false".

Guarda anche

SymbolInfoSessionQuote, Symbol info, TimeToStruct, Date structure

CustomSymbolSetSessionTrade

Imposta l'ora di inizio e fine della sessione di trading specificata per il simbolo e il giorno della settimana specificati.

```
bool  CustomSymbolSetSessionTrade(
   const string      symbol_name,          // nome simbolo
   ENUM_DAY_OF_WEEK  day_of_week,          // giorno della settimana
   uint              session_index,        // indice della sessione
   datetime          from,                 // orario d'inizio della sessione
   datetime          to                    // orario di fine della sessione
   );
```

Parametri

symbol_name

[in] Nome simbolo personalizzato.

ENUM_DAY_OF_WEEK

[in] Giorno della settimana, valore dall'enumerazione ENUM_DAY_OF_WEEK.

uint

[in] Indice della sessione, per il quale devono essere impostate le ore di inizio e fine. L'indicizzazione della sessione inizia da 0.

from

[in] Orario di inizio sessione in secondi da 00:00, il valore dati della variabile viene ignorato.

to

[in] Orario di fine sessione in secondi da 00:00, il valore dei dati nella variabile viene ignorato.

Valore di Ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Se la sessione con lo specificato *session_index* già esiste, la funzione modifica semplicemente l'inizio e la fine della sessione.

Se sono stati passati i parametri di partenza e di fine, zero, per la sessione, (*from=0* e *to=0*), la sessione appropriata con il *session_index* viene eliminata, mentre l'indicizzazione della sessione viene spostata verso il basso.

Le sessioni possono essere aggiunte solo sequenzialmente. In altre parole, puoi aggiungere *session_index=1* solo se la sessione con l'indice 0 esiste già. Se questa regola è interrotta, non viene creata una nuova sessione, e la funzione restituisce "false".

Guarda anche

SymbolInfoSessionTrade, Symbol info, TimeToStruct, Date structure

CustomRatesDelete

Elimina tutte le barre dallo storico dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato.

```
int  CustomRatesDelete(
   const string     symbol,        // nome simbolo
   datetime         from,          // inizio data
   datetime         to             // fine data
   );
```

Parametri

symbol

[in] Nome simbolo personalizzato.

from

[in] Orario della prima barra dello storico dei prezzi entro l'intervallo specificato da rimuovere.

to

[in] Orario dell'ultima barra nello storico dei prezzi entro l'intervallo specificato da rimuovere.

Valore di Ritorno

Numero di barre cancellate o -1 in caso di un errore.

Guarda anche

CustomRatesReplace, CustomRatesUpdate, CopyRates

CustomRatesReplace

Sostituisce completamente lo storico dei prezzi del simbolo personalizzato entro l'intervallo di tempo specificato con i dati dell' array di tipo MqlRates.

```
int  CustomRatesReplace(
   const string     symbol,              // nome del simbolo
   datetime         from,                // data d'inizio
   datetime         to,                  // data di fine
   const MqlRates&  rates[],             // array per i dati da applicare al simbolo pe
   uint             count=WHOLE_ARRAY    // numero degli elementi dell'array rates[] da
   );
```

Parametri

symbol

[in] Nome simbolo personalizzato.

from

[In] Orario della prima barra nella cronologia dei prezzi entro l'intervallo specificato da aggiornare.

to

[in] Orario dell'ultima bar nello storico dei prezzi entro l'intervallo specificato da aggiornare.

rates[]

[in] Array dei dati storico di tipo MqlRates per M1.

Valore di Ritorno

Numero di barre aggiornate o -1 in caso di errore.

Nota

Se la barra dall'array *rates[]* va oltre l'intervallo specificato, viene ignorata. Se una tale barra è già presente nello storico dei prezzi e rientra nell'intervallo specificato, viene sostituita. Tutte le altre barre dell'attuale storico dei prezzi al di fuori dell'intervallo specificato restano invariate. L'array dati *rates[]* dovrebbe essere corretto per quanto riguarda i prezzi OHLC, mentre i tempi di apertura della barra dovrebbero corrispondere al timeframe M1.

Guarda anche

CustomRatesDelete, CustomRatesUpdate, CopyRates

CustomRatesUpdate

Aggiunge barre mancanti allo storico dei simboli personalizzati e sostituisce i dati esistenti con quelli dell'array di tipo MqlRates.

```
int  CustomRatesUpdate(
   const string     symbol,             // nome del simbolo personalizzato
   const MqlRates&  rates[],            // array per i dati da applicare al simbolo pe
   uint             count=WHOLE_ARRAY   // numero degli elementi dell'array rates[] da
   );
```

Parametri

symbol

[in] Nome simbolo personalizzato.

rates[]

[in] Array dei dati di storico di tipo MqlRates per M1.

count=WHOLE_ARRAY

[in] Numero di elementi dell'array *rates[]* da utilizzare per l'aggiornamento. WHOLE_ARRAY significa che devono essere usati tutti gli elementi dell'array *rates[]*.

Valore di Ritorno

Numero di barre aggiornate o -1 in caso di errore.

Nota

Se non esiste un barra dall'array *rates[]* nello storico del simbolo personalizzato corrente, viene aggiunta. Se tale barra esiste già, viene sostituita. Tutte le altre barre dello storico dei prezzi corrente, rimangono invariate. L'array dati *rates[]* dovrebbe essere corretto per quanto riguarda i prezzi OHLC, mentre i tempi di apertura della barra dovrebbero corrispondere al timeframe M1.

Guarda anche

CustomRatesReplace, CustomRatesDelete, CopyRates

CustomTicksAdd

Aggiunge i dati da un array di tipo MqlTick allo storico dei prezzi di un simbolo personalizzato. Il simbolo personalizzato deve essere selezionato nella finestra del Watch Market.

```
int  CustomTicksAdd(
   const string     symbol,              // Nome del simbolo
   const MqlTick&   ticks[],             // L'array con i dati tick che devono essere a
   uint             count=WHOLE_ARRAY    // numero degli elementi dell'array ticks[] da
   );
```

Parametri

symbol

[in] Il nome del simbolo personalizzato.

ticks[]

[in] Un array di dati tick di tipo MqlTick arrangiati in ordine di tempo da dati più vecchi a quelli più recenti, cioè ticks[k].time_msc <= ticks[n].time_msc, if k<n.

count=WHOLE_ARRAY

[in] Numero di elementi dell'array *ticks[]* da utilizzare per l'aggiunta. WHOLE_ARRAY significa che devono essere usati tutti gli elementi dell'array *ticks[]* .

Valore di Ritorno

Il numero di ticks aggiunti o -1 in caso di errore.

Ulteriori Note

La funzione CustomTicksAdd funziona solo per i simboli personalizzati aperti nella finestra del Watch Market. Se il simbolo non è selezionato nel Market Watch, è necessario aggiungere ticks utilizzando CustomTicksReplace.

La funzione CustomTicksAdd consente di trasmettere i ticks come se fossero stati consegnati dal server del broker. I dati vengono inviati alla finestra di Market Watch anziché essere scritti direttamente nel database dei tick. Il terminale quindi salva i ticks dal Market Watch in un database. Se la quantità di dati trasmessa durante la chiamata di funzione è ampia, il comportamento della funzione cambia in modo da ridurre l'utilizzo delle risorse. Se si passano più di 256 ticks, i dati vengono divisi in due parti. La prima, cioè la parte più grande, viene scritta direttamente sul database dei tick (come avviene in CustomTicksReplace). La seconda parte contenente 128 ticks viene passata alla finestra Market Watch, da cui il terminal salva i ticks nel database.

La struttura MqlTick ha due campi con il valore temporale: time (il tempo tick in secondi) e il time_msc (il tempo tick in millisecondi) che vengono contati dal 1° gennaio 1970. Questi campi nei codici aggiunti vengono elaborati nel seguente ordine:

1. Se ticks[k].time_msc!=0, lo usiamo per riempire il campo ticks[k].time, cioè ticks[k].time=ticks[k].time_msc/1000 (divisione integer) viene impostato per il tick
2. Se ticks[k].time_msc==0 e ticks[k].time!=0, il tempo in millisecondi viene ottenuto moltiplicando per 1000, cioè ticks[k].time_msc=ticks[k].time*1000
3. If ticks[k].time_msc==0 e ticks[k].time==0, viene scritto in questi campi il corrente orario del trade server fino ai millisecondi come da momento della chiamata CustomTicksAdd.

Se il valore di ticks[k].bid, ticks[k].ask, ticks[k].last or ticks[k].volume, è più grande di zero, una combinazione di flags viene scritta nel campo ticks[k].flags:

- TICK_FLAG_BID – il tick ha cambiato il prezzo bid
- TICK_FLAG_ASK – il tick ha cambiato il prezzo ask
- TICK_FLAG_LAST – il tick ha cambiato il prezzo last deal
- TICK_FLAG_VOLUME – il tick ha cambiato volume

Se il valore di un campo è inferiore o uguale a zero, la corrispondente flag non viene scritta nel campo ticks[k].flags.

Flags TICK_FLAG_BUY e TICK_FLAG_SELL non vengono aggiunte allo storico di un simbolo personalizzato.

Guarda anche

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

CustomTicksDelete

Elimina tutti i ticks dalla cronologia dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato.

```
int  CustomTicksDelete(
   const string     symbol,            // nome del simbolo
   long             from_msc,          // data inizio
   long             to_msc             // data fine
   );
```

Parametri

symbol

[in] Nome simbolo personalizzato.

from_msc

[in] Orario del primo tick nello storico dei prezzi entro l'intervallo specificato da rimuovere. Orario in millisecondi dal 01.01.1970.

to_msc

[in] Orario dell'ultima cifra nello storico dei prezzi entro l'intervallo specificato da rimuovere. Orario in millisecondi dal 01.01.1970.

Valore di Ritorno

Numero di ticks cancellati o -1 in caso di un errore.

Guarda anche

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

CustomTicksReplace

Sostituisce completamente lo storico dei prezzi del simbolo personalizzato entro l'intervallo di tempo specificato con i dati dall'array di tipo MqlTick.

```
int  CustomTicksReplace(
   const string     symbol,            // nome del simbolo
   long             from_msc,          // data d'inizio
   long             to_msc,            // data di fine
   const MqlTick&   ticks[],           // array per i dati da applicare al periodo pe
   uint             count=WHOLE_ARRAY  // numero degli elementi dell'array tick[] da
   );
```

Parametri

symbol

[in] Nome simbolo personalizzato.

from_msc

[in] Orario del primo tick nello storico dei prezzi entro l'intervallo specificato da rimuovere. Orario in millisecondi dal 01.01.1970.

to_msc

[in] Orario dell'ultima cifra nello storico dei prezzi entro l'intervallo specificato da rimuovere. Orario in millisecondi dal 01.01.1970.

ticks[]

[in] Array dati tick di tipo MqlTick ordinato per orario in ordine ascendente.

count=WHOLE_ARRAY

[in] Numero di elementi dell'array *ticks[]* da utilizzare per la sostituzione nell'intervallo di tempo specificato. WHOLE_ARRAY significa che devono essere usati tutti gli elementi dell'array *ticks[]* .

Valore di Ritorno

Numero di ticks aggiornati o -1 in caso di errore.

Nota

Poiché diversi ticks spesso possono avere lo stesso orario fino ad un millisecondo in un flusso di quotazioni (l'orario accurato dei tick è memorizzato nel campo *time_msc* della struttura MqlTick), la funzione CustomTicksReplace non ordina automaticamente gli elementi dell'array *ticks[]* per orario. Pertanto, l'array di ticks deve essere pre-arrangiato in ordine ascendente di tempo.

I ticks vengono sostituiti consecutivamente, giorno dopo giorno, fino al momento specificato in *to_msc* o fino all'avvenimento di un errore. Viene elaborato il primo giorno dall'intervallo specificato seguito da quello successivo, ecc. Non appena viene rilevata la mancata corrispondenza tra l'ora del tick e l'ordine ascendente (non-discendente), la sostituzione del tick si arresta sul giorno corrente. Tutti i ticks dei giorni precedenti vengono sostituiti con successo, mentre il giorno corrente (al momento di un tick sbagliato) e tutti i giorni rimanenti nell'intervallo specificato rimangono invariati.

Se l'array *ticks[]* non contiene dati per qulsiasi giorno (in genere, in qualsiasi intervallo di tempo), un "buco" corrispondente ai dati mancanti viene visualizzato nello storico dei simboli personalizzati

dopo che i dati tick da `ticks[]` vengono applicati. In altre parole, la chiamata di CustomTicksReplace con ticks mancanti equivale a cancellare una parte dello storico dei tick, come se venisse chiamato CustomTicksDelete con l'intervallo "buco".

Se il database tick non dispone di dati per l'intervallo di tempo specificato, CustomTicksReplace aggiungerà al database i ticks dall'array ticks[].

La funzione CustomTicksReplace funziona direttamente con il database dei tick.

Guarda anche

CustomRatesDelete, CustomRatesUpdate, CustomTicksDelete, CopyTicks, CopyTicksRange

CustomBookAdd

Passa lo status del Depth of Market per un simbolo personalizzato. La funzione consente di trasmettere il Depth of Market come se i prezzi arrivassero dal server di un broker.

```
bool  CustomBookAdd(
   const string        symbol,            // nome del simbolo
   const MqlBookInfo&  books[]            // array con la descrizione degli elementi d
   uint                count=WHOLE_ARRAY  // numero di elementi da usare
   );
```

Parametri

simbolo

[in] Nome simbolo personalizzato.

books[]

[in] L'array di tipo di dati MqlBookInfo che descrivono completamente lo status del Depth of Market — tutte le richieste di buy e sell. Lo status del Depth of Market passato sostituisce completamente il precedente.

count=WHOLE_ARRAY

[in] Il numero di elementi dell'array 'books' da passare alla funzione. L'intero array viene utilizzato per impostazione predefinita.

Valore di ritorno

true - successo, altrimenti - false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

La funzione CustomBookAdd funziona solo per i simboli personalizzati per cui viene aperto il Depth of Market — tramite l'interfaccia della piattaforma o la funzione MarketBookAdd.

Quando si lancia il Depth of Market, i prezzi Bid e Ask del simbolo non vengono aggiornati. Dovresti controllare il cambiamento dei prezzi migliori e lanciare i ticks usando CustomTicksAdd.

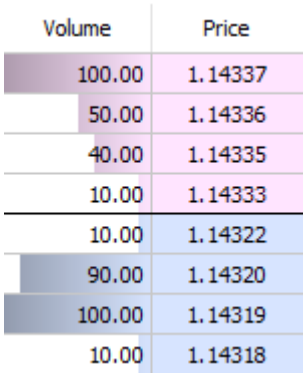
La funzione verifica l'esattezza dei dati trasmessi: per ogni elemento deve essere indicato il tipo, il prezzo e il volume. Inoltre, MqlBookInfo.volume e MqlBookInfo.volume_real non devono essere zero o negativi; se entrambi i volumi sono negativi, questo sarà considerato un errore. Puoi specificare uno qualsiasi dei volumi o entrambi — verrà utilizzato quello indicato o positivo:

```
volume=-1 && volume_real=2 — verrà utilizzato volume_real=2,
volume=3 && volume_real=0 — verrà utilizzato volume=3.
```

Quando si salvano i dati, viene verificato il parametro "Profondità del Book" (SYMBOL_TICKS_BOOKDEPTH) del simbolo personalizzato destinatario. Se il numero di richieste di Sell supera questo valore nella Depth of Market passato, i livelli in eccesso vengono scartati. Lo stesso vale per le richieste di Buy.

Esempio di riempimento dell'array "books":

Status del Depth of Market	Riempimento books[]

	books[0].type=BOOK_TYPE_SELL;
	books[0].price=1.14337;
	books[0].volume=100;
	books[1].type=BOOK_TYPE_SELL;
	books[1].price=1.14330;
	books[1].volume=50;
	books[2].type=BOOK_TYPE_SELL;
	books[2].price=1.14335;
	books[2].volume=40;
	books[3].type=BOOK_TYPE_SELL;
	books[3].price=1.14333;
	books[3].volume=10;
	books[4].type=BOOK_TYPE_BUY;
	books[4].price=1.14322;
	books[4].volume=10;
	books[5].type=BOOK_TYPE_BUY;
	books[5].price=1.14320;
	books[5].volume=90;
	books[6].type=BOOK_TYPE_BUY;
	books[6].price=1.14319;
	books[6].volume=100;
	books[7].type=BOOK_TYPE_BUY;
	books[7].price=1.14318;
	books[7].volume=10;

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione inizializzazione expert                                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ailita il Depth of Market per un simbolo da cui prendiamo i dati
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione expert                               |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Funzione Tick                                                    |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
```

```
//--- copia i prezzi correnti dal simbolo comune a quello personalizzato
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| Funzione Book                                                    |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- copia lo stato attuale della Profondità di mercato dal simbolo comune a quello p
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
//+------------------------------------------------------------------+
```

Guarda anche

MarketBookAdd, CustomTicksAdd, OnBookEvent

Operazioni col Chart

Le funzioni per l'impostazione delle proprietà del chart (ChartSetInteger, ChartSetDouble, ChartSetString) sono asincrone e vengono utilizzate per inviare comandi di aggiornamento al chart. Se queste funzioni vengono eseguite correttamente, il comando viene incluso nella coda comune degli eventi del chart. Le modifiche alle proprietà del chart vengono implementate insieme alla gestione della coda degli eventi di questo chart.

Pertanto, non aspettarsi un aggiornamento immediato del chart dopo aver chiamato le funzioni asincrone. Usare la funzione ChartRedraw () per aggiornare forzatamente l'aspetto e le proprietà del chart.

Funzione	Azione
ChartApplyTemplate	Applica un modello(template) specifico da un file specificato al chart
ChartSaveTemplate	Salva le impostazioni correnti chart in un template con un nome specificato
ChartWindowFind	Restituisce il numero di una sottofinestra in cui è disegnato un indicatore
ChartTimePriceToXY	Converte le coordinate di un chart dalla rappresentazione tempo/prezzo alle coordinate X e Y
ChartXYToTimePrice	Converte le coordinate X e Y su un chart dei valori di tempo e di prezzo
ChartOpen	Apre un nuovo chart con il simbolo e periodo specificato
ChartClose	Chiude il chart specificato
ChartFirst	Restituisce l'ID del primo chart del terminale client
ChartNext	Restituisce l'ID del chart successivo a quello specificato
ChartSymbol	Restituisce il nome del simbolo del chart specificato
ChartPeriod	Restituisce il valore del periodo del chart specificato
ChartRedraw	Chiama un ridisegno forzato di un chart specificato
ChartSetDouble	Imposta il valore double per una proprietà corrispondente del chart specificato
ChartSetInteger	Imposta il valore integer (datetime, int, color, bool o char) per una proprietà corrispondente del chart specificato
ChartSetString	Imposta il valore di stringa per una proprietà corrispondente del chart specificato
ChartGetDouble	Restituisce il valore double della proprietà del chart specificato
ChartGetInteger	Restituisce il valore integer della proprietà del chart specificato
ChartGetString	Restituisce il valore string della proprietà del chart specificato

Funzione	Azione
ChartNavigate	Esegue lo slittamento del chart indicato, per il numero specificato di barre rispetto alla posizione specificata nel chart
ChartID	Restituisce l'ID del chart corrente
ChartIndicatorAdd	Aggiunge un indicatore con l'handle specificato in una finestra del chart specificato
ChartIndicatorDelete	Rimuove un indicatore con un nome specificato dalla finestra del chart specificato
ChartIndicatorGet	Restituisce l'handle dell'indicatore con il nome breve specificato nella finestra del chart specificato
ChartIndicatorName	Restituisce il nome breve dell'indicatore per numero nella lista indicatori sulla finestra del chart specificato
ChartIndicatorsTotal	Restituisce il numero di tutti gli indicatori applicati alla finestra del chart specificato.
ChartWindowOnDropped	Restituisce il numero (indice) della sottofinestra chart, a cui è stato allegato l'Expert Advisor o Script.
ChartPriceOnDropped	Restituisce la coordinata prezzo del punto nel chart, a cui è stato allegato l'Expert Advisor o Script.
ChartTimeOnDropped	Restituisce la coordinata temporale del punto nel chart, a cui è stato allegato l'Expert Advisor o lo Script.
ChartXOnDropped	Restituisce la coordinata X del punto del chart, a cui è stato allegato l' Expert Advisor o lo Script.
ChartYOnDropped	Restituisce la coordinata Y del punto del chart, a cui è stato allegato l' Expert Advisor o lo Script.
ChartSetSymbolPeriod	Cambia il valore di simbolo e periodo di chart specificato
ChartScreenShot	Fornisce uno screenshot del chart del suo stato attuale in formato GIF, PNG o BMP a seconda dell'estensione specificata

ChartApplyTemplate

Applica un modello specifico da un file specificato al grafico. Il comando viene aggiunto alla coda dei messaggi del chart e verrà eseguito dopo l'elaborazione di tutti i comandi precedenti.

```
bool  ChartApplyTemplate(
   long          chart_id,      // Chart ID
   const string  filename       // Nome file Template
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

filename

[in] Il nome del file contenente il template.

Valore restituito

Restituisce true se il comando è stato aggiunto alla coda del chart, altrimenti false. Per avere informazioni sull' errore, Chiamare la funzione GetLastError().

Nota

L'Expert Advisor verrà de-caricato e non sarà in grado di continuare ad operare in caso di caricamento di successo di un nuovo template per il grafico a cui è allegato.

When applying the template to the chart, trade permissions may be limited due to security reasons:

> **Live trading permission cannot be extended for the Expert Advisors launched by applying the template using ChartApplyTemplate() function.**

If the mql5-program calling ChartApplyTemplate() function has no permission to trade, the Expert Advisor launched via the template will also not be able to trade regardless of the template settings.

If the mql5-program calling ChartApplyTemplate() function has permission to trade, while there is no such permission in the template settings, the Expert Advisor launched via the template will not be able to trade.

Utilizzo dei modelli

Le risorse del linguaggio MQL5 permettono di impostare più proprietà del grafico, compresi i colori che utilizzano la funzione ChartSetInteger():

- Colore di sfondo del grafico;
- Colore degli assi, la scala e la linea OHLC;
- Colore della griglia;
- Colore dei volumi e dei livelli di apertura delle posizioni;
- Colore della barra superiore, ombra e bordo della candela rialzista(_*bullish);
- Colore della barra inferiore, ombra e bordo della candela ribassista(_*bearish);
- Colore della linea del grafico e candele Doji;

- Colore del corpo della candela rialzista(_*bullish);
- Colore del corpo della candela ribassista(_*bearish);
- Colore della linea di prezzo di Offerta(_*Bid);
- Colore della linea prezzo di Domanda(_*Ask);
- Colore della linea del prezzo ultumo affare (Last);
- Colore dei livelli stop order (Stop Loss e Take Profit).

Inoltre, ci possono essere molteplici oggetti grafici ed indicatori su un grafico. È possibile impostare un grafico con tutti gli indicatori necessari, e poi salvarlo come template. Tale template può essere applicato a qualsiasi grafico.

La funzione ChartApplyTemplate() è intesa all' utilizzo di un template precedentemente salvato, e può essere utilizzata in qualsiasi programma MQL5. Il percorso del file che memorizza il template è passato come secondo parametro di ChartApplyTemplate(). Il file template viene cercato in base alle seguenti regole:

- se il separatore barra rovesciata(backslash) "\" (scritto come "\\") è posto all'inizio del percorso, il template è cercato relativo al percorso _terminal_data_directory\MQL5,,
- se non c'è backslash, il template viene cercato per il relativo eseguibile file EX5, in cui ChartApplyTemplate() viene chiamato;
- se un template non si trova nelle prime due varianti, la ricerca viene eseguita nella cartella terminal_directory\Profiles\Templates\.

Qui terminal_directory è la cartella da cui MetaTrader 5 Terminal Client è in esecuzione, e terminal_data_directory è la cartella in cui sono memorizzati i file modificabili, la sua posizione dipende dal sistema operativo, il nome utente e le impostazioni di sicurezza del computer. Normalmente sono cartelle diverse, ma in alcuni casi possono coincidere.

La posizione delle cartelle e terminal_data_directory terminal_directory può essere ottenuta usando la funzione TerminalInfoString().

```
//--- directory da cui viene avviato il terminale
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("Terminal directory:",terminal_path);
//--- terminale directory di dati, in cui si trova la cartella MQL5 con EA ed indicat
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Terminal data directory:",terminal_data_path);
```

Ad esempio:

```
//--- per un template in terminal_data_directory\MQL5\
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- ricerca per un template nella directory_di_EX5_file\, poi in terminal_data_direc
ChartApplyTemplate(0,"second_template.tpl"))
//--- ricerca per un template nella directory_di_EX5_file\Miei_templates\, poi nella
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

I templates non sono risorse, non possono essere inclusi in un file eseguibile EX5 file.

Esempio:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- example of applying template, located in \MQL5\Files
   if(FileIsExist("my_template.tpl"))
     {
      Print("The file my_template.tpl found in \Files'");
      //--- apply template
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("The template 'my_template.tpl' applied successfully");
         //--- redraw chart
         ChartRedraw();
        }
      else
         Print("Failed to apply 'my_template.tpl', error code ",GetLastError());
     }
   else
     {
      Print("File 'my_template.tpl' not found in "
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

Vedi anche

Risorse

ChartSaveTemplate

Salva le impostazioni correnti del grafico in un template con un nome specificato.

```
bool  ChartSaveTemplate(
   long          chart_id,     // Chart ID
   const string  filename      // Nomefile per salvare un template
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

filename

[in] Il nome del file per salvare il template. L'estensione ".tpl" verrà aggiunta al nome del file automaticamente, non c'è bisogno di specificarla. Il template viene salvato in **directory_terminale**\Profiles\Templates\ e può essere utilizzato per l'applicazione manuale nel terminale. Se un template con lo stesso nome esiste già, il contenuto di questo file verrà sovrascritto.

Valore restituito

In caso di successo, la funzione restituisce true, altrimenti restituisce false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Usando i templates, è possibile salvare le impostazioni di grafico con tutti gli indicatori ed oggetti grafici applicati, per poi applicarli ad un altro grafico.

Esempio:

```
//+------------------------------------------------------------------------------------+
//|                                                    Test_ChartSaveTemplate.mq5 |
//|                                 Copyright 2011, MetaQuotes Software Corp. |
//|                                                             https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- parametri di input
input string               symbol="GBPUSD";  // Il simbolo di un nuovo grafico
input ENUM_TIMEFRAMES      period=PERIOD_H3; // Il timeframe di un nuovo grafico
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Per prima cosa alleghiamo gli indicatori al grafico
   int handle;
```

```
//--- Prepariamo l'indicatore per l'uso
   if(!PrepareZigzag(NULL,0,handle)) return; // Fallimento, quindi uscita
//--- Alleghiamo l'indicatore al grafico corrente, ma in una finestra separata.
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("Fallimento nell'allegare al grafico %s/%s l'indicatore con l'handle
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- Termina operazione del programma
      return;
     }
//--- Aggiorna il del grafico per visualizzare l'indicatore
   ChartRedraw();
//--- Trova le ultime due fratture del zig-zag
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("Fallimento nel trovare le ultime due fratture del ZigZag!");
      //--- Termina operazione del programma
      return;
     }
//--- Ora colleghiamo un canale di deviazione standard
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("Fallimento nel creare l'oggetto %s. Codice errore %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- Il canale è stato creato, definiamo il secondo punto
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- Imposta un testo tooltip per il canale
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo dall' MQL5 Help");
      //--- Aggiorna il grafico
      ChartRedraw();
     }
//--- Salva il risultato in un template
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- Apre un nuovo grafico ed applica un template salvato ad esso
   long new_chart=ChartOpen(symbol,period);
   //--- Abilita i suggerimenti per gli oggetti grafici
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
```

```
      //--- Applica il template salvato al grafico
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------------------+
//| Crea un handle a zig-zag e garantisce la disponibilità dei suoi dati         |
//+------------------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- L'indicatore Zigzag deve essere situato in terminal_data_folder\MQL5\Examples
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: Fallimento nel creare l' handle dell' indicatore ZigZag. Codice
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- Quando si crea un handle di indicatore, esso richiede tempo per calcolare i valo
   int k=0; // Il numero di tentativi di attesa per il calcolo dell'indicatore
//--- Attende per il calcolo in un ciclo, fermandosi a 50 millisecondi se il calcolo n
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- Mostra il numero di tentativi
      PrintFormat("%s: k=%d",__FUNCTION__,k);
      //--- Aspettare 50 millisecondi di attesa fino a quando l'indicatore viene calco
      Sleep(50);
      //--- Se più di 100 tentativo, allora c'è qualcosa che non va
      if(k>100)
        {
         //--- Riporta un problema
         PrintFormat("Fallimento nel calcolo dell' indicatore per %d tentativi!");
         //--- Termina operazione del programma
         return false;
        }
     }
//--- Tutto è pronto, l'indicatore viene creato ed i valori sono calcolati
   return true;
  }
//+------------------------------------------------------------------------------+
//| Cerca le ultime 2 fratture zigzag e le piazza negli array                     |
//+------------------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];          // Un array per i valori del zigzag
   datetime times[];         // Un array per ottenere l'orario
   int size=100;             // Grandezza dell'array
```

```
   ResetLastError();
//--- Copia gli ultimi 100 valori dell'indicatore
   int copied=CopyBuffer(handle,0,0,size,values);
//--- Controlla il numero di valori copiati
   if(copied<100)
     {
      PrintFormat("%s: Fallimento nel copiare %d valori dell' indicatore con l'handle=
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- Definisce l'ordine di accesso alla matrice come in una timeseries
   ArraySetAsSeries(values,true);
//--- Scrivi qui il numero di barre, in cui le fratture sono state trovate
   int positions[];
//--- Imposta la grandezza dell'array
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- Contatori
   int i=0,k=0;
//--- Avvia la ricerca delle fratture
   while(i<100)
     {
      double v=values[i];
      //--- Non siamo interessati a valori vuoti
      if(v!=0.0)
        {
         //--- Ricorda il numero della barra
         positions[k]=i;
         //--- Ricorda il valore del zigzag sulla frattura
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- Incrementa il contatore
         k++;
         //--- Se vengono trovate due fratture, interrompere il ciclo
         if(k>2) break;
        }
      i++;
     }
//--- Definisce l'ordine di accesso agli array come in una timeseries
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: Fallimento nella copia di %d valori da CopyTime(). Codice erro
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- Apre l'orario di apertura della barra, in cui le ultime 2 fratture si sono veri
   get_times[0]=times[positions[1]];// Il penultimo valore, sarà scritto come la prim
   get_times[1]=times[positions[2]];// Il terzultimo valore sarà la seconda frattura
   PrintFormat("%s: first=%s,  second=%s",__FUNCTION__,TimeToString(get_times[1]),Time
```

```
//--- Successo
   return true;
  }
```

Vedi anche

ChartApplyTemplate(), Resources

ChartWindowFind

La funzione restituisce il numero di una sottofinestra dove un indicatore viene disegnato. Ci sono due varianti della funzione.

1. La funzione ricerca nel grafico indicato per la sottofinestra con il "nome breve" specificato dell' indicatore (il nome breve è mostrato nella parta superiore sinistra della sottofinestra) e restitusce il numero della sottofinestra in caso di sucesso.

```
int  ChartWindowFind(
   long     chart_id,                  // identificatore del grafico
   string   indicator_shortname        // nome breve dell'indicatore, vedere INDICATOR
```

2. La funzione dev'essere chiamata dall'indicatore personalizzato. Restituisce il numero della sottofinestra dove l'indicatore sta lavorando.

```
int  ChartWindowFind();
```

Parametri

chart_id

[in] ID del Grafico. 0 indica il grafico corrente.

indicator_shortname

[in] Nome breve dell'indicatore.

Valore restituito

Numero della sottofinestra in caso di successo. In caso di fallimento, la funzione restituisce -1In case of failure the function returns -1.

Nota

Se la seconda variante della funzione (senza parametri) è chiamata da uno Script o Expert Advisor, la funzione restituisce -1.

Non mischiare il nome breve dell'indicatore ed il nome del file, che viene specificato quando un indicatore viene creato usando le funzioni iCustom() ed IndicatorCreate(). Se il nome breve dell'indicatore non è impostato esplicitamente, allora il nome del file contenente il codice sorgente dell'indicatore è specificato in esso durante la compilazione.

E' importante formare correttamente il nome breve dell'indicatore, che è registrato nella proprietà INDICATOR_SHORTNAME usando la funzione IndicatorSetString(). E' raccomandabile che il nome breve contenga valori dei parametri di input dell'indicatore, perchè l'indicatore cancellato dal grafico nella funzione ChartIndicatorDelete() viene identificato dal suo nome breve.

Esempio:

```
#property script_show_inputs
//--- parametri di input
input string   shortname="MACD(12,26,9)";
//+------------------------------------------------------------------------------+
//| Restituisce il numero delle finestra del chart con questo indicatore         |
//+------------------------------------------------------------------------------+
```

```
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
      //--- la funzione viene chiamata dall'inicatore, il nome non è richiesto
      window=ChartWindowFind();
     }
   else
     {
      //--- la funzione viene chiamata dall'Expert Advisor o Script
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortname);
   if(window!=-1)
      Print("Indicator "+shortname+" is in the window #"+(string)window);
   else
      Print("Indicator "+shortname+" is not found. window = "+(string)window);
  }
```

Vedi anche

ObjectCreate(), ObjectFind()

ChartTimePriceToXY

Converte le coordinate di un grafico da una rappresentazione tempo/prezzo nelle coordinate X ed Y.

```
bool  ChartTimePriceToXY(
   long           chart_id,      // ID del Grafico
   int            sub_window,    // Il numero di sottofinestra
   datetime       time,          // Orario sul grafico
   double         price,         // Prezzo sul grafico
   int&           x,             // Le coordinate X per l'orario sul grafico
   int&           y              // Le coordinate Y per l'orario sul grafico
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

sub_window

[in] Il numero della sottofinestra grafico. 0 significa la finestra principale del grafico.

time

[in] Il valore temporale sul grafico, per cui il valore in pixel, lungo l'asse X, verrà ricevuto. L'origine è l'angolo superiore sinistro della finestra principale del grafico.

price

[in] Il valore del prezzo sul grafico, per cui il valora in pixel lungo l'asse Y verrà ricevuto. L'origine è l'angolo superiore sinistro della finestra principale del grafico.

x

[out] La variabile, in cui la conversione del tempo ad X verrà ricevuta.

y

[out] La variabile, in cui la conversione del prezzo ad Y verrà ricevuta.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Vedi anche

ChartXYToTimePrice()

ChartXYToTimePrice

Converte le coordinate X ed Y sul grafico, nei valori tempo e prezzo.

```
bool  ChartXYToTimePrice(
   long            chart_id,      // ID del Grafico
   int             x,             // La coordinata X sul grafico
   int             y,             // La coordinata Y sul grafico
   int&            sub_window,    // Il numero di sottofinestra
   datetime&       time,          // Orario sul grafico
   double&         price          // Prezzo sul grafico
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

x

[in] La coordinata X .

y

[in] La coordinata Y.

sub_window

[out] La variabile, in cui verrà scritto il numero della sottofinestra del grafico. 0 significa la finestra principale del grafico.

time

[out] Il valore temporale sul grafico, per cui il valore in pixel lungo l'asse X verrà ricevuto. L'origine è l'angolo superiore sinistro della finestra principale del grafico.

price

[out] Il valore prezzo sul grafico, per cui il valore in pixel lungo l'asse Y verrà ricevuto. L'origine è l'angolo superiore sinistro della finestra principale del grafico.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Esempio:

```
//+------------------------------------------------------------------------------------+
//| Funzione ChartEvent                                                                |
//+------------------------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Mostra i parametri dell'evento sul chart
```

```
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spa
//--- Se questo è un evento di un click del mouse sul chart
   if(id==CHARTEVENT_CLICK)
     {
      //--- Prepara le variabili
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- Converte le coordinate X ed Y in termini di data/orario
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Window=%d X=%d  Y=%d  =>  Orario=%s  Prezzo=%G",window,x,y,Time
         //--- Esegue la conversione rovesciata: (X,Y) => (Time,Price)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,
         else
            Print("ChartTimePriceToXY return error code: ",GetLastError());
         //--- delete lines
         ObjectDelete(0,"V Line");
         ObjectDelete(0,"H Line");
         //--- create horizontal and vertical lines of the crosshair
         ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
         ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
         ChartRedraw(0);
        }
      else
         Print("ChartXYToTimePrice return error code: ",GetLastError());
      Print("+------------------------------------------------------------------+");
     }
  }
```

Vedi anche

ChartTimePriceToXY()

ChartOpen

Apre un nuovo grafico con il simbolo e periodo specificati.

```
long  ChartOpen(
   string           symbol,     // Nome del Simbolo
   ENUM_TIMEFRAMES  period      // Periodo
   );
```

Parametri

symbol

[in] Simbolo del Grafico. NULL significa il simbolo del grafico corrente (a cui è allegato l' Expert Advisor).

period

[in] Periodo del Grafico (timeframe). Può essere uno dei valori ENUM_TIMEFRAMES. 0 indica il corrente periodo del grafico.

Valore restituito

In caso di successo, restituisce l'ID del grafico aperto. Altrimenti restituisce 0.

Nota

Il numero massimo possibile di grafici aperti simultaneamente nel terminale non può superare il valore CHARTS_MAX.

ChartFirst

Restituisce l'ID del primo grafico del terminale client.

```
long  ChartFirst();
```

Valore restituito

Chart ID.

ChartNext

Restituisce l'ID del Grafico successivo a quello specificato

```
long  ChartNext(
   long  chart_id      // ID del Grafico
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 non significa il grafico corrente. 0 significa "restituisce l' ID del primo grafico".

Valore restituito

ID del Grafico. Se questa è la fine dell'elenco del grafico, restituisce -1.

Esempio:

```
//--- variabili per l'ID del grafico
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst =",ChartSymbol(prevChart)," ID =",prevChart);
   while(i<limit)// Abbiamo certamente non più di 100 grafici aperti
     {
      currChart=ChartNext(prevChart); // Ottiene l'ID del nuovo grafico usando l'ID de
      if(currChart<0) break;          // Ha raggiunto la fine della lista dei grafici
      Print(i,ChartSymbol(currChart)," ID =",currChart);
      prevChart=currChart;// salviamo l'ID del grafico corrente per ChartNext()
      i++;// Non dimentichiamo di incrementare il contatore
     }
```

ChartClose

Chiude il grafico specificato.

```
bool  ChartClose(
   long  chart_id=0      // ID del Grafico
   );
```

Parametri

chart_id=0

[in] ID del Grafico. 0 significa il grafico corrente.

Valore restituito

In caso di successo, restituisce true, altrimenti false.

ChartSymbol

Restituisce il nome del simbolo per il grafico specificato.

```
string  ChartSymbol(
   long  chart_id=0      // ID del Grafico
   );
```

Parametri

chart_id=0

[in] ID del Grafico. 0 significa il grafico corrente.

Valore restituito

Se il grafico non esiste, il risultato sarà una stringa vuota.

Vedi anche

ChartSetSymbolPeriod

ChartPeriod

Restituisce il periodo timeframe del grafico specificato.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // ID del Grafico
   );
```

Parametri

chart_id=0

[in] ID del Grafico. 0 significa il grafico corrente.

Valore restituito

La funzione restituisce uno dei valori ENUM_TIMEFRAMES. Se il grafico non esiste, restituisce 0.

ChartRedraw

Questa funzione chiama un ridisegno forzato di un grafico specificato.

```
void  ChartRedraw(
   long  chart_id=0      // ID del Grafico
   );
```

Parametri

chart_id=0

[in] ID del Grafico. 0 significa il grafico corrente.

Nota

Generalmente viene utilizzato dopo la modifica della proprietà di oggetti.

Vedi anche

Funzioni Oggetti

ChartSetDouble

Imposta un valore per la corrispondente proprietà del grafico specificato. La proprietà del chart dev' essere di tipo double. Il comando viene aggiunto alla coda dei messaggi del chart e verrà eseguito dopo l'elaborazione di tutti i comandi precedenti.

```
bool  ChartSetDouble(
   long                          chart_id,      // ID del Grafico
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // ID della Proprietà
   double                        value          // Valore
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

prop_id

[in] ID proprietà del Grafico. Puo essere uno dei valori ENUM_CHART_PROPERTY_DOUBLE (eccetto le proprietà di sola-lettura).

valore

[in] Valore proprietà.

Valore restituito

Restituisce true se il comando è stato aggiunto alla coda del chart, altrimenti false. Per avere informazioni sull' errore, Chiamare la funzione GetLastError().

Note

La funzione è asincrona, il che significa che la funzione non attende l'esecuzione del comando, che è stato correttamente aggiunto alla coda di specifica del chart. Invece, restituisce immediatamente il controllo. La proprietà cambierà solo dopo la gestione del comando appropriato dalla coda del chart. Per eseguire immediatamente i comandi dalla coda del chart, chiamare la funzione ChartRedraw.

Se si desidera modificare immediatamente più proprietà del chart contemporaneamente, allora le funzioni corrispondenti (ChartSetString, ChartSetDouble, ChartSetString) devono essere eseguite in un blocco di codice, dopo il quale è necessario chiamare ChartRedraw una volta.

Per verificare il risultato dell'esecuzione del comando, è possibile utilizzare una funzione che richiede la proprietà del chart specificata (ChartGetInteger, ChartGetDouble, ChartSetString). Tuttavia, notare che queste funzioni sono sincrone e attendere i risultati dell'esecuzione.

ChartSetInteger

Imposta un valore per la corrispondente proprietà del chart specificato. La proprietà del chart dev'essere datetime, int, color, bool or char. Il comando viene aggiunto alla coda dei messaggi del chart e verrà eseguito dopo l'elaborazione di tutti i comandi precedenti.

```
bool  ChartSetInteger(
   long                          chart_id,      // ID del Chart
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // ID della Proprietà
   long                          value          // Valore
   );
```

```
bool  ChartSetInteger(
   long                          chart_id,      // ID del Chart
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // ID della Proprietà
   int                           sub_window,    // Numero della Sottofinestra
   long                          value          // Valore
   );
```

Parametri

chart_id

[in] ID del Chart. 0 significa il chart corrente.

prop_id

[in] ID proprietà del Chart. Può essere uno dei valori ENUM_CHART_PROPERTY_INTEGER (tranne le proprietà di sola-lettura).

sub_window

[in] Numero di sottofinestra chart. Per il primo caso, il valore di default è 0 (finestra chart principale). La maggior parte delle proprietà non richiedono un numero sottofinestra.

valore

[in] Valore proprietà.

Valore restituito

Restituisce true se il comando è stato aggiunto alla coda del chart, altrimenti false. Per avere informazioni sull' errore, chiamare la funzione GetLastError().

Note

La funzione è asincrona, il che significa che la funzione non attende l'esecuzione del comando, che è stato correttamente aggiunto alla coda di specifica del chart. Invece, restituisce immediatamente il controllo. La proprietà cambierà solo dopo la gestione del comando appropriato dalla coda del chart. Per eseguire immediatamente i comandi dalla coda del chart, chiamare la funzione ChartRedraw.

Se si desidera modificare immediatamente più proprietà del chart contemporaneamente, allora le funzioni corrispondenti (ChartSetString, ChartSetDouble, ChartSetString) devono essere eseguite in un blocco di codice, dopo il quale è necessario chiamare ChartRedraw una volta.

Per verificare il risultato dell'esecuzione del comando, è possibile utilizzare una funzione che richiede la proprietà del chart specificata (ChartGetInteger, ChartGetDouble, ChartSetString). Tuttavia, notare che queste funzioni sono sincrone e attendere i risultati dell'esecuzione.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione inizializzazione Expert                                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Abilitazione degli eventi dei movimenti del mouse sulla finestra del chart
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- L'aggiornamento forzato delle proprietà del chart garantisce la preparazione pe
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse sinstra
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse deastra
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse centrale
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse primo tasto X
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse secondo tasto X
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // tasto shift
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // tasto control
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

ChartSetString

Imposta un valore per la corrispondente proprietà del grafico specificato. La proprietà del grafico dev'essere di tipo stringa. Il comando viene aggiunto alla coda dei messaggi del chart e verrà eseguito dopo l'elaborazione di tutti i comandi precedenti.

```
bool  ChartSetString(
   long                        chart_id,    // ID del Grafico
   ENUM_CHART_PROPERTY_STRING  prop_id,     // ID della Proprietà
   string                      str_value    // Valore
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

prop_id

[in] ID proprietà del Grafico. Il suo valore può essere uno dei valori ENUM_CHART_PROPERTY_STRING (eccetto le proprietà di sola-lettura).

str_value

[in] Valore proprietà stringa. La lunghezza della stringa non può eccedere i 2045 caratteri (i caratteri extra verranno troncati).

Valore restituito

Restituisce true se il comando è stato aggiunto alla coda del chart, altrimenti false. Per avere informazioni sull' errore, Chiamare la funzione GetLastError().

Nota

ChartSetString può essere usato per un output commento sul grafico al posto della funzione Comment.

La funzione è asincrona, il che significa che la funzione non attende l'esecuzione del comando, che è stato correttamente aggiunto alla coda di specifica del chart. Invece, restituisce immediatamente il controllo. La proprietà cambierà solo dopo la gestione del comando appropriato dalla coda del chart. Per eseguire immediatamente i comandi dalla coda del chart, chiamare la funzione ChartRedraw.

Se si desidera modificare immediatamente più proprietà del chart contemporaneamente, allora le funzioni corrispondenti (ChartSetString, ChartSetDouble, ChartSetString) devono essere eseguite in un blocco di codice, dopo il quale è necessario chiamare ChartRedraw una volta.

Per verificare il risultato dell'esecuzione del comando, è possibile utilizzare una funzione che richiede la proprietà del chart specificata (ChartGetInteger, ChartGetDouble, ChartSetString). Tuttavia, notare che queste funzioni sono sincrone e attendere i risultati dell'esecuzione.

Esempio:

```
void OnTick()
  {
//---
   double Ask,Bid;
```

```
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
   string comment=StringFormat("Выводим цены:\nAsk = %G\nBid = %G\nSpread = %d",
                               Ask,Bid,Spread);
   ChartSetString(0,CHART_COMMENT,comment);
  }
```

Vedi anche

Comment, ChartGetString

ChartGetDouble

Restituisce il valore di una proprietà corrispondente del chart specificato. La proprietà del chart dev' essere di tipo double. Ci sono 2 varianti di chiamate di funzione.

1. Restituisce il valore della proprietà direttamente.

```
double  ChartGetDouble(
   long                          chart_id,         // ID del Chart
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // ID della Proprietà
   int                           sub_window=0      // numero della sottofinestra, se nec
   );
```

2. Restituisce true o false, a seconda se la funzione ha avuto successo. In caso di successo, il valore della proprietà viene posto in una variabile target double_var passata per riferimento.

```
bool  ChartGetDouble(
   long                          chart_id,         // ID del Chart
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // ID della Proprietà
   int                           sub_window,       // Numero della Sottofinestra
   double&                       double_var        // Variabile Target per la proprietà
   );
```

Parametri

chart_id

[in] ID del Chart. 0 significa il chart corrente.

prop_id

[in] ID proprietà del Chart. Questo valore può essere uno dei valori ENUM_CHART_PROPERTY_DOUBLE.

sub_window

[in] Numero di sottofinestra chart. Per il primo caso, il valore di default è 0 (finestra chart principale). La maggior parte delle proprietà non richiedono un numero sottofinestra.

double_var

[out] Variabile target di tipo double per la proprietà richiesta.

Valore restituito

Il valore di tipo double.

Per il caso della seconda chiamata restituisce true se la proprietà specificata è disponibile es il suo valore è stato inserito nella variabile double_var, altrimenti restituisce false. Per avere ulteriori informazioni sull' errore, è necessario richiamare la funzione GetLastError().

Nota

La funzione è sincrona, il che significa che attende l'esecuzione di tutti i comandi che sono stati aggiunti alla coda del chart prima della sua chiamata.

Esempio:

```
voidOnStart()
  {
   double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
   double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
   Print("CHART_PRICE_MIN =",priceMin);
   Print("CHART_PRICE_MAX =",priceMax);
  }
```

ChartGetInteger

Restituisce il valore di una proprietà corrispondente del grafico specificato. La proprietà del grafico deve essere di tipo datetime, int o bool. Ci sono 2 varianti di chiamate di funzione.

1. Restituisce il valore della proprietà direttamente.

```
long  ChartGetInteger(
   long                          chart_id,         // ID del Grafico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // ID della Proprietà
   int                           sub_window=0      // numero della sottofinestra, se ne
   );
```

2. Restituisce true o false, a seconda se la funzione ha avuto successo. In caso di successo, il valore della proprietà viene posto in una variabile target long_var passata per riferimento.

```
bool  ChartGetInteger(
   long                          chart_id,         // ID del Grafico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // ID della Proprietà
   int                           sub_window,       // numero sottofinestra
   long&                         long_var          // Variabile target per la proprietà
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

prop_id

[in] ID proprietà del Grafico. Questo valore può essere uno dei valori ENUM_CHART_PROPERTY_INTEGER.

sub_window

[in] Numero di sottofinestra grafico. Per il primo caso, il valore di default è 0 (finestra grafico principale). La maggior parte delle proprietà non richiedono un numero sottofinestra.

long_var

[out] Variabile target di tipo long per la proprietà richiesta.

Valore restituito

Il valore di tipo long.

Per il caso della seconda chiamata restituisce true se la proprietà specificata è disponibile ed il suo valore è stato memorizzato nella variabile long_var, altrimenti restituisce false. Per ottenere ulteriori informazioni sull' errore, è necessario richiamare la funzione GetLastError().

Nota

La funzione è sincrona, il che significa che attende l'esecuzione di tutti i comandi che sono stati aggiunti alla coda del chart prima della sua chiamata.

Esempio:

```
voidOnStart()
  {
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   Print("CHART_HEIGHT_IN_PIXELS =",height,"pixels");
   Print("CHART_WIDTH_IN_PIXELS =",width,"pixels");
  }
```

ChartGetString

Restituisce il valore di una proprietà corrispondente del grafico specificato. Proprietà del grafico deve essere di tipo stringa. Ci sono 2 varianti della chiamata di funzione.

1. Restituisce il valore della proprietà direttamente.

```
string  ChartGetString(
   long                        chart_id,         // ID del Grafico
   ENUM_CHART_PROPERTY_STRING  prop_id           // ID della Proprietà
   );
```

2. Restituisce true o false, a seconda se la funzione ha avuto successo. In caso di successo, il valore della proprietà viene posto in una variabile target string_var passata per riferimento.

```
bool  ChartGetString(
   long                        chart_id,         // ID del Grafico
   ENUM_CHART_PROPERTY_STRING  prop_id,          // ID della Proprietà
   string&                     string_var        // Variable target per la proprietà
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

prop_id

[in] ID proprietà del Grafico. Questo valore può essere uno dei valori ENUM_CHART_PROPERTY_STRING.

string_var

[out] Variabile target di tipo stringa per la proprietà richiesta.

Valore restituito

Il valore di tipo stringa.

Per il caso della seconda chiamata restituisce true se la proprietà specificata è disponibile ed il suo valore è stato memorizzato nella variabile string_var, altrimenti restituisce false. Per ottenere ulteriori informazioni sull' errore, è necessario richiamare la funzione GetLastError().

Nota

ChartGetString può essere usato per leggere commenti riportato nel diagramma utilizzando le funzioni Comment o ChartSetString.

La funzione è sincrona, il che significa che attende l'esecuzione di tutti i comandi che sono stati aggiunti alla coda del chart prima della sua chiamata.

Esempio:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
```

```
   ChartRedraw();
   Sleep(1000);
   string comm=ChartGetString(0,CHART_COMMENT);
   Print(comm);
  }
```

Vedi anche

Comment, ChartSetString

ChartNavigate

Esegue lo slittamento del grafico indicato dal numero specificato di barre rispetto alla posizione specificata nel grafico.

```
bool  ChartNavigate(
   long                  chart_id,      // ID del Grafico
   ENUM_CHART_POSITION   position,      // Posizione
   int                   shift=0        // Valore di Slittamento
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

position

[in] Posizione Grafico per eseguire uno slittamento. Può essere uno dei valore ENUM_CHART_POSITION.

shift=0

[in] Numero di barre per slittare il grafico. Il valore positivo indica lo slittamento a destra (fino alla fine del grafico), il valore negativo indica lo slittamento a sinistra (all'inizio del grafico). Lo slittamento zero può essere utilizzato per spostarsi all'inizio o alla fine del grafico.

Valore restituito

Restituisce true se ha successo, altrimenti restituisce false.

Esempio:

```
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                             |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- ottiene l'handle del grafico corrente
   long handle=ChartID();
   string comm="";
   if(handle>0) // se con successo, addizionalmente imposta il grafico
     {
      //--- disabilita auto scroll
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- imposta uno slittamento dal bordo destro del grafico
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- disegna candele
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- impostare la modalità di visualizzazione per i volumi tick
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

      //--- prepara il testo per l'output in Comment()
      comm="Slitta di 10 barre a destra dell'inizio dall' inizio dello storico";
```

```
      //--- mostra commento
      Comment(comm);
      //--- slitta di 10 barre a destra dell'inizio dall' inizio dello storico
      ChartNavigate(handle,CHART_BEGIN,10);
//--- ottiene il numero della prima barra visibile sul grafico (numerazione come nelle
      long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      //--- aggiunge carattere di avanzamento riga
      comm=comm+"\r\n";
      //--- aggiunge al commento
      comm=comm+"La prima barra nel grafico è il numero "+IntegerToString(first_bar)+"
      //--- mostra commento
      Comment(comm);
      //--- attendere 5 secondi per vedere come si muove il grafico
      Sleep(5000);

      //--- aggiunge al commento di testo
 comm = comm +"\\ R \\ n"+"Slitta 10 barre a sinistra del bordo grafico a destra";
      Comment(comm);
/ / --- Slitta 10 barre a sinistra del bordo grafico a destra
      ChartNavigate(handle,CHART_END,-10);
//--- ottiene il numero della prima barra visibile sul grafico (numerazione come nelle
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"La prima barra nel grafico è il numero "+IntegerToString(first_bar)+"
      Comment(comm);
      //--- attendere 5 secondi per vedere come si muove il grafico
      Sleep(5000);

      //--- nuovo blocco di slittamento del grafico
      comm=comm+"\r\n"+"Scroll 300 bars to the right of the history start";
      Comment(comm);
      //--- slitta di 300 barre alla destra dell' inizio dello storico
      ChartNavigate(handle,CHART_BEGIN,300);
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"La prima barra nel grafico è il numero "+IntegerToString(first_bar)+"
      Comment(comm);
      //--- attendere 5 secondi per vedere come si muove il grafico
      Sleep(5000);

      //--- nuovo blocco di slittamento del grafico
      comm=comm+"\r\n"+"Slitta di 300 barre a sinistra del bordo grafico a destra";
      Comment(comm);
      //--- slitta di 300 barre a sinistra del bordo grafico a destra
      ChartNavigate(handle,CHART_END,-300);
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"La prima barra nel grafico è il numero "+IntegerToString(first_bar)+"
      Comment(comm);
```

```
      }
   }
```

ChartID

Restituisce l'ID del grafico corrente.

```
long  ChartID();
```

Valore restituito

Valore di tipo long.

ChartIndicatorAdd

Aggiunge un indicatore con l'handle specificato in una finestra di grafico specificato. Indicatore e grafico dovrebbero essere generati sullo stesso simbolo e timeframe.

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // ID del Grafico
   int   sub_window                 // numero di sotto-finestre
   int   indicator_handle           // handle dell'indicatore
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

sub_window

[in] Il numero del grafico sotto-finestra. 0 significa la finestra principale del grafico. Per aggiungere un indicatore in una nuova finestra, il parametro deve esserne uno maggiore rispetto all'indice dell'ultima finestra esistente, cioè uguale a CHART_WINDOWS_TOTAL. Se il valore del parametro è maggiore di CHART_WINDOWS_TOTAL, una nuova finestra non verrà creata, e l'indicatore non verrà aggiunto.

indicator_handle

[in] Handle dell'indicatore.

Valore restituito

La funzione restituisce true in caso di successo, altrimenti restituisce false. Al fine di ottenere informazioni sull' errore, chiamare la funzione GetLastError(). Errore 4114 significa che un grafico ed un indicatore aggiunto differiscono dal loro simbolo o timeframe.

Nota

Se un indicatore che deve essere disegnato in una sottofinestra separata (per esempio, il iMACD built-in o un indicatore personalizzato con specificata la proprietà #property indicator_separate_window) viene applicato alla finestra grafico principale, può non essere visibile anche se sarà ancora presente nella lista degli indicatori. Ciò significa che la scala dell'indicatore è diversa dalla scala del grafico dei prezzi ed i valori applicati agli indicatori non possono essere inseriti nel campo visualizzato del grafico dei prezzi. In questo caso, GetLastError() restituisce codice zero, indicando l'assenza di un errore. I valori di tale indicatore "invisibile" possono essere visti nella Finestra Dati e ricevuti da altre applicazioni MQL5.

Esempio:

```
#property description "L' Expert Advisor dimostra il lavoro con la funzione ChartIndic
#property description "Dopo il lancio sul chart (e la ricezione dell'errore nel Journa
#property description "le proprietà dell'Expert Advisor e specificare correttamente i
#property description "L' indicatore MACD verrà aggiunto al chart."

//--- parametri di input
input string          symbol="AUDUSD";     // nome del simbolo
input ENUM_TIMEFRAMES period=PERIOD_M12;   // time frame
```

```
input int    fast_ema_period=12;          // periodo MACD veloce
input int    slow_ema_period=26;          // periodo MACD lento
input int      signal_period=9;           // periodo del segnale
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // tipo di prezzo per il calcolo del MACD

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- prova ad aggiungere l'indicatore sul grafico
   if(!AddIndicator())
     {
      //--- La funzione AddIndicator() rifiuta di aggiungere l'indicatore sul chart
      int answer=MessageBox("Vuoi aggiungere MACD sul chart ad ogni modo?",
                            "Non corretto simbolo e/o timeframe per aggiungere l'indic
                            MB_YESNO // Le selezioni dei bottoni "Si" e "No" verranno
                            );
      //--- se un utente insiste ancora sull'uso non corretto di ChartIndicatorAdd()
      if(answer==IDYES)
        {
         //--- prima di tutto, verrà creata una riga del Journal a riguardo
         PrintFormat("Attenzione! %s: Si sta tentando di aggiungere l'indicatore MACD
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //--- riceve il numero di una nuova sottofinestra, alla quale tenteremo di ag
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- ora fa un tentativo destinato al fallimento
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("Fallimento nell'aggiungere l'indicatore MACD sulla finestra c
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                    |
//+------------------------------------------------------------------------------+
void OnTick()
  {
// L' Expert Advisor non esegue nulla
  }
//+------------------------------------------------------------------------------+
//| La funzione per controllare ed aggiungere l'indicatore sul chart             |
//+------------------------------------------------------------------------------+
bool AddIndicator()
```

```
  {
//--- messaggio visualizzato
   string message;
//--- controlla se il simbolo dell'indicatore ed il simbolo del chart corrispondono l
   if(symbol!=_Symbol)
     {
      message="Visualizza l'uso della funzione Demo_ChartIndicatorAdd():";
      message=message+"\r\n";
      message=message+"Impossibile aggiungere l'indicatore calcolato su un altro simbo
      message=message+"\r\n";
      message=message+"Specifica il simbolo del chart nelle proprietà dell'Expert Advi
      Alert(message);
      //--- uscita prematura, l'indicatore non verrà aggiunto sul chart
      return false;
     }
//--- controlla se l'indicatore ed il timeframe del chart corrispondono l'un l'altro
   if(period!=_Period)
     {
      message="Impossibile aggiungere l'indicatore calcolato su un altro timeframe sul
      message=message+"\r\n";
      message=message+"Specifica il timeframe del chart nelle proprietà dell'Expert Ad
      Alert(message);
      //--- uscita prematura, l'indicatore non verrà aggiunto sul chart
      return false;
     }
//--- tutti i controlli completati, simbolo e timeframe dell'indicatore corrispondono
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__,"  Crea l'indicatore MACD ");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("Fallimento nel creare l'indicatore MACD. Error code ",GetLastError());
        }
     }
//--- resetta il codice errore
   ResetLastError();
//--- applica l'indicatore al chart
   Print(__FUNCTION__,"  Aggiunge l'indicatore MACD sul chart");
   Print("MACD viene generato su ",symbol,"/",EnumToString(period));
//--- riceve il numero di una nuova sottofinestra, a cui viene aggiunto l'indicatore M
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("Aggiunta dell'indicatore MACD sulla finestra chart chart %d",subwindow
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("Fallimento nell'aggiungere l'indicatore MACD sulla finestra del cha
                  subwindow,GetLastError());
     }
//--- Indicatore aggiunto con successo
```

```
   return(true);
  }
```

Vedere anche

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

ChartIndicatorDelete

Rimuove un indicatore con il nome specificato dalla finestra del grafico specificato.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // ID del Grafico
   int            sub_window             // numero della sottofinestra
   const string   indicator_shortname    // nome breve dell'indicatore
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 indica il grafico corrente.

sub_window

[in] Numero di sottofinestra grafico. 0 denota la principale sottofinestra del grafico.

const indicator_shortname

[in] Nome breve dell'indicatore che si trova nella proprietà INDICATOR_SHORTNAME con la funzione IndicatorSetString(). Per ottenere il nome breve di un indicatore utilizzare la funzione ChartIndicatorName().

Valore restituito

Restituisce true in caso di eliminazione di successo dell'indicatore. In caso contrario, restituisce false. Per ottenere i dettagli dell' errore utilizzare la funzione GetLastError().

Nota

Se due indicatori con identici nomi brevi esistono nella sottofinestra del grafico, il primo in riga verrà eliminato.

Se altri indicatori su questo grafico si basano sui valori dell'indicatore che viene eliminato, tali indicatori verranno cancellati.

Non confondere il nome breve indicatore e il nome del file che viene specificato durante la creazione di un indicatore con le funzioni iCustom() ed IndicatorCreate(). Se il nome breve di un indicatore non è impostato in modo esplicito, il nome del file che contiene il codice sorgente dell'indicatore verrà specificato durante la compilazione.

L' eliminazione di un indicatore da un grafico non vuol dire che la sua parte di calcolo sarà cancellata dalla memoria del terminale. Per rilasciare l'handle utilizzare la funzione IndicatorRelease().

Il nome breve dell'indicatore dovrebbe essere formato correttamente. Questo viene scritto nella proprietà INDICATOR_SHORTNAME utilizzando la funzione IndicatorSetString(). E' raccomandato che il nome breve deve contenere i valori di tutti i parametri di input dell'indicatore, poiché l'indicatore eliminato dal grafico dalla funzione ChartIndicatorDelete() è identificato dal nome breve.

Esempio di eliminazione di un indicatore dopo l'inizializzazione non riuscito:

```
//+------------------------------------------------------------------------------+
//|                                                 Demo_ChartIndicatorDelete.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
```

```
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int       first_param=1;
input int       second_param=2;
input int       third_param=3;
input bool      wrong_init=true;
//--- buffers indicatore
double          HistogramBuffer[];
string          shortname;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- Collega l'array HistogramBuffer al buffer indicatore
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- Costruisce un nome indicatore corto basato sui parametri di input
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- Se viene impostato il completamento forzato di un indicatore, restituisce un va
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
                const long &volume[],
                const int &spread[])
  {
//--- Posizione d'inizio per lavorare in loop
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- Compila il buffer di indicatore con valori
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=close[i];
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Un handler dell'evento Deinit                                                |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: Codice del motivo della Deinizializzazione=%d",__FUNCTION__,reason
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("Un indicatore con il nome corto %s (file %s) elimina se stesso dal
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- Analizza il risultato della chamata di ChartIndicatorDelete()
      if(!res)
        {
         PrintFormat("Fallimento nell'eliminare l'indicatore %s dalla finestra #%d. Co
                     shortname,window,GetLastError());
        }
     }
  }
```

Vedi anche

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

ChartIndicatorGet

Restituisce l'handle dell'indicatore con il nome breve specificato nella finestra del grafico specificato.

```
int  ChartIndicatorGet(
   long           chart_id,              // ID del Grafico
   int            sub_window             // Il numero di sottofinestra
   const string   indicator_shortname    // Nome breve dell'indicatore
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

sub_window

[in] Il numero della sottofinestra grafico. 0 significa la finestra principale del grafico.

const indicator_shortname

[in] Il nome breve se l'indicatore, che si trova nella proprietà INDICATOR_SHORTNAME utilizza la funzione IndicatorSetString(). Per ottenere il nome breve di un indicatore, utilizzare la funzione ChartIndicatorName().

Valore restituito

Restituisce un handle indicatore in caso di successo, altrimenti restituisce INVALID_HANDLE. Per ottenere informazioni sull' errore, chiamare la funzione GetLastError().

Nota

L'handle dell'indicatore ottenuto utilizzando la funzione ChartIndicatorGet() incrementa il contatore dell'utilizzo interno dell'indicatore. Il sistema di runtime del terminale mantiene tutti gli indicatori, il cui contatore è maggiore di zero, caricati. Pertanto, l' handle dell'indicatore che non è più necessario deve essere rilasciato immediatamente ed esplicitamente utilizzando IndicatorRelease() nello stesso programma che lo ha ricevuto, come mostrato nell'esempio seguente. Altrimenti, sarà impossibile trovare l'handle "abbandonato" e rilasciarlo correttamente da un altro programma.

Durante la creazione di un indicatore, fare attenzione a formare il suo nome breve, che è scritto nella proprietà INDICATOR_SHORTNAME utilizzando la funzione IndicatorSetString(). Si raccomanda che un nome breve deve contenere i valori dei parametri di ingresso dell'indicatore, poiché l'indicatore è identificato nella funzione ChartIndicatorGet() in base al suo nome breve.

Un altro modo per identificare l'indicatore è quello di ottenere una lista dei suoi parametri per una handle dato, utilizzando la funzione IndicatorParameters() e quindi analizzare i valori ottenuti.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   //--- Il numero di finestre sul chart (almeno una finestra principale è sempre pres
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
```

```
   //--- Controlla tutte le finestre
   for(int w=0;w<windows;w++)
     {
      //--- il numero di indicatori in questa finestra/sottofinestra
      int total=ChartIndicatorsTotal(0,w);
      //--- Va attraverso tutti gli indicatori nella finestra
      for(int i=0;i<total;i++)
        {
         //--- ottiene il nome corto dell'indicatore
         string name=ChartIndicatorName(0,w,i);
         //--- ottiene l'handle dell'indicatore
         int handle=ChartIndicatorGet(0,w,name);
         //--- Aggiunge al log
         PrintFormat("Finestra=%d,  indice=%d,  nome=%s,  handle=%d",w,i,name,handle);
         //--- Devi obbligatoriamente rilasciare l' handle dell'indicatore quando non
         IndicatorRelease(handle);
        }
     }
  }
```

Vedi anche

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

ChartIndicatorName

Restituisce il nome breve dell'indicatore per il numero nella lista degli indicatori sulla finestra del grafico specificato.

```
string  ChartIndicatorName(
   long  chart_id,       // ID del Grafico
   int   sub_window      // numero di sottofinestra
   int   index           // Indice dell'indicatore nell'elenco degli indicatori aggiunt
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 indica il grafico corrente.

sub_window

[in] Numero di sottofinestra grafico. 0 denota la principale sottofinestra del grafico.

index

[in] l'indice dell'indicatore nell'elenco degli indicatori. La numerazione degli indicatori inizia con zero, vale a dire che il primo indicatore della lista ha indice 0. Per ottenere il numero di indicatori nella lista utilizzare la funzione ChartIndicatorsTotal().

Valore restituito

Il nome breve dell'indicatore che si trova nella proprietà INDICATOR_SHORTNAME con la funzione IndicatorSetString(). Per ottenere i dettagli dell'errore utilizzare la funzione GetLastError().

Nota

Non confondere il nome breve indicatore e il nome del file che viene specificato durante la creazione di un indicatore con le funzioni iCustom() ed IndicatorCreate(). Se il nome breve di un indicatore non è impostato in modo esplicito, il nome del file che contiene il codice sorgente dell'indicatore verrà specificato durante la compilazione.

Il nome breve dell'indicatore dovrebbe essere formato correttamente. Questo viene scritto nella proprietà INDICATOR_SHORTNAME utilizzando la funzione IndicatorSetString(). E' raccomandato che il nome breve deve contenere i valori di tutti i parametri di input dell'indicatore, poiché l'indicatore eliminato dal grafico dalla funzione ChartIndicatorDelete() è identificato dal nome breve.

Vedi anche

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

ChartIndicatorsTotal

Restituisce il numero di tutti gli indicatori applicati alla finestra del grafico specificato.

```
int  ChartIndicatorsTotal(
   long  chart_id,       // ID del Grafico
   int   sub_window      // numero di sottofinestra
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 indica il grafico corrente.

sub_window

[in] Numero di sottofinestra grafico. 0 denota la principale sottofinestra del grafico.

Valore restituito

Il numero di indicatori nella finestra del grafico specificato. Per ottenere i dettagli dell' errore utilizzare la funzione GetLastError().

Nota

La funzione permette di fare la ricerca attraverso tutti gli indicatori collegati al grafico. Il numero di tutte le finestre del grafico può essere ottenuto dalla proprietà CHART_WINDOWS_TOTAL utilizzando la funzione ChartGetInteger().

Vedi anche

ChartIndicatorAdd(), ChartIndicatorDelete(), iCustom(), IndicatorCreate(), IndicatorSetString()

ChartWindowOnDropped

Restituisce il numero (index) della sottofinestra del grafico, a cui è allegato l'Expert Avisor o lo Script. 0 significa la finestra principale del grafico.

```
int  ChartWindowOnDropped();
```

Valore restituito

Value of int type.

Esempio:

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("Lo script sta girano sulla finestra #"+myWindow+
         ". Total windows on the chart "+ChartSymbol()+":",windowsTotal);
```

Vedi anche

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

ChartPriceOnDropped

Restituisce le coordinate prezzo corrispondenti al punto del grafico, dove cui un Expert Advisor o Script è stato allegato.

```
double  ChartPriceOnDropped();
```

Valore restituito

Valore di tipo double.

Esempio:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

Vedi anche

ChartXOnDropped, ChartYOnDropped

ChartTimeOnDropped

Restituisce le coordinate temporali corrispondenti del punto del grafico, a cui è stato allegato l'Expert Advisor.

```
datetime  ChartTimeOnDropped();
```

Valore restituito

Value of datetime type.

Esempio:

```
   datetime t=ChartTimeOnDropped();
   Print("Lo script era allegato su"+t);
```

Vedi anche

ChartXOnDropped, ChartYOnDropped

ChartXOnDropped

Restituisce la coordinata X del punto del grafico, a cui è stato allegato un Expert Advisor o Script.

```
int  ChartXOnDropped();
```

Valore restituito

Il valore della coordinata X.

Nota

Direzione dell'asse X da sinistra a destra.

Esempio:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

Vedi anche

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

ChartYOnDropped

Restituisce le coordinate Y del punto del grafico, a cui è stato allegato un Expert Advisor o Script..

```
int  ChartYOnDropped();
```

Valore restituito

Il valore della coordinata Y.

Nota

Direzione dell'asse Y dall'alto al basso.

Vedi anche

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

ChartSetSymbolPeriod

Cambia il simbolo ed il periodo per il grafico specificato. La funzione è asincrona, cioè invia il comando e non aspetta per il suo completamento. Il comando viene aggiunto alla coda dei messaggi del chart e verrà eseguito dopo l'elaborazione di tutti i comandi precedenti.

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // ID del Grafico
   string           symbol,        // Nome del Simbolo
   ENUM_TIMEFRAMES  period         // Periodo
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

symbol

[in] Simbolo del Grafico. Il valore NULL significa che il corrente simbolo del grafico (a cui l'Expert Advisor è allegato)

period

[in] Periodo del Grafico (timeframe). Può essere uno dei valori ENUM_TIMEFRAMES. 0 indica il corrente periodo del grafico.

Valore restituito

Restituisce true se il comando è stato aggiunto alla coda del chart, altrimenti false. Per avere informazioni sull' errore, Chiamare la funzione GetLastError().

Nota

Il cambio di simbolo/periodo porta alla re-inizializzazione dell' Expert Advisor allegato al grafico.

La chiamata di ChartSetSymbolPeriod con lo stesso simbolo e timeframe può essere utilizzata per aggiornare il grafico (simile al comando Aggiorna del terminale). A sua volta, l'aggiornamento del chart innesca ri-calcolo degli indicatori ad esso collegati. Così, è possibile calcolare un indicatore sul grafico, anche se non ci sono ticks (ad esempio, nei fine settimana).

Vedi anche

ChartSymbol, ChartPeriod

ChartScreenShot

La funzione fornisce uno screenshot del grafico nel suo stato corrente nel formato GIF, PNG o BMP a seconda estensione specificata.

```
bool  ChartScreenShot(
   long             chart_id,                  // ID del Grafico
   string           filename,                  // Nome del Simbol
   int              width,                     // Larghezza
   int              height,                    // Altezza
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // Tipo di Allineamento
   );
```

Parametri

chart_id

[in] ID del Grafico. 0 significa il grafico corrente.

filename

[in] Il nome del file Screenshot. Non può superare i 63 caratteri. I files di Screenshot si trovano nella directory \Files.

width

[in] Larghezza Screenshot in pixels.

height

[in] Altezza Screenshot in pixels.

align_mode=ALIGN_RIGHT

[in] Output mode di uno screenshot sottile. Valore dell'enumerazione ENUM_ALIGN_MODE . ALIGN_RIGHT significa allinea al margine destro (l'output dalla fine). ALIGN_LEFT significa justifica a Sinistra.

Valore restituito

Restituisce true se ha successo, altrimenti false.

Nota

Se avete bisogno di prendere uno screenshot da un grafico a partire da una certa posizione, in primo luogo è necessario posizionare il grafico utilizzando la funzione ChartNavigate(). Se la dimensione orizzontale della schermata è più piccola della finestra del grafico, sia la parte destra della finestra del grafico, che la sua parte sinistra, vengono date in output a seconda delle impostazioni align_mode.

Esempio:

```
#property description "L'Expert Advisor dimostra come creare una serie di immagini del
#property description "grafico usando la funzione ChartScreenShot(). Per comodità, il
#property descrizione "mostrata sul grafico. L'altezza e la larghezza delle immagini è

#define        WIDTH  800     // Larghezza immagine per chiamare ChartScreenShot()
#define        HEIGHT 600     // Altezza immagine per chiamare ChartScreenShot()
```

```
//--- parametri di input
input int      pictures=5;    // Il numero di immagini nella serie
int            mode=-1;       // -1 denota uno slittamento al lato destro del grafico,
int            bars_shift=300;// In numero di barre quando si slitta il grafico usando
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                          |
//+------------------------------------------------------------------------------------+
void OnInit()
  {
//--- Disattiva auto-scorrimento del grafico
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- Imposta lo slittamento del bordo destro del grafico
   ChartSetInteger(0,CHART_SHIFT,true);
//--- Mostra un grafico a candela
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("La preparazione dell'Expert Advisor è completa");
  }
//+------------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                          |
//+------------------------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------------------------+
//| Funzione ChartEvent                                                                |
//+------------------------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//---Mostra il nome della funzione, orario di chiamata ed identificatore vento
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- Maneggia l'evento CHARTEVENT_CLICK ("Un click del mouse sul grafico")
   if(id==CHARTEVENT_CLICK)
     {
      //--- Slittamento iniziale dal lato del grafico
      int pos=0;
      //--- Operazione con il lato sinistro del grafico
      if(mode>0)
        {
         //--- Slitam il grafico al lato sinistro
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
```

```
            //--- Prepara il testo da mostrare sul grafico ed il nome file
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- Mostra il nome sul grafico come un commento
            Comment(name);
            //--- Salva lo screenshot del grafico in un file interminal_directory\MQL5
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("Abbiamo salvato lo screenshot ",name);
            //---
            pos+=bars_shift;
            //--- Da all'utente il tempo di guardare alla nuova parte del grafico
            Sleep(3000);
            //--- Scorre il grafico dalla posizione corrente barre bars_shift a destra
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- Cambia la modalità nell'opposto
         mode*=-1;
        }
      else // Operazione con il lato destro del grafico
        {
         //--- Scorre il grafico al lato destro
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- Prepara il testo da mostrare sul grafico ed il nome file
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- Mostra il nome sul grafico come un commento
            Comment(name);
            //--- Salva lo screenshot del grafico in un file interminal_directory\MQL5
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("Abbiamo salvato lo screenshot ",name);
            //---
            pos+=bars_shift;
            //--- Da all'utente il tempo di guardare alla nuova parte del grafico
            Sleep(3000);
            //--- Scorre il grafico dalla posizione corrente barre bars_shift a destra
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- Cambia la modalità nell'opposto
         mode*=-1;
        }
     }  // Cerca l'event handling CHARTEVENT_CLICK
//--- Fine dell' handler OnChartEvent()
  }
```

Vedi anche

ChartNavigate(), Risorse

Funzioni di Trade

Questo è il gruppo di funzioni intese a gestire le attività di trading.

Prima di procedere a studiare le funzioni di trade della piattaforma, devi avere una chiara comprensione dei termini di base: ordine(order), affare(o accordo, deal) e posizione(position).

- Unordine è un'istruzione che viene data a un broker per acquistare o vendere uno strumento finanziario. Ci sono due principali tipi di ordini: Mercato(market) e Pendente(pending). Inoltre, ci sono speciali livelli Take Profit e Stop Loss.
- Unaffare è lo scambio commerciale (acquisto o vendita) di una garanzia finanziaria. L'acquisto viene eseguito al prezzo di domanda (Ask) e la vendita viene eseguita al prezzo di offerta (Bid). Un affare può essere aperto a seguito dell'esecuzione dell'ordine di mercato o dell'attivazione dell'ordine pendente. Si noti che in alcuni casi l'esecuzione di un ordine può comportare numerosi affari.
- Una posizione é un obbligo di trade, vale a dire il numero di contratti acquistati o venduti di uno strumento finanziario. Una posizione long è la sicurezza finanziaria acquistata in attesa che il prezzo della sicurezza aumenti. Una posizione short è l'obbligo di fornire una sicurezza che prevede che il prezzo scenderà in futuro.

Generiche informazioni sulle operazioni di trading sono disponibili nella guida del terminale client.

Funzioni di trading possono essere utilizzate in Expert Advisor e scripts. Funzioni di trading possono essere chiamate solo se nelle proprietà della Expert Advisor o dello script, l'opzione "Consenti trading live" è abilitata.

Trading can be allowed or prohibited depending on various factors described in the Trade Permission section.

Funzione	Azione
OrderCalcMargin	Calcola il margine richiesto per il tipo di ordine specificato, nella valuta di deposito
OrderCalcProfit	Calcola il profitto in base ai parametri passati, nella valuta di deposito
OrderCheck	Verifica se ci sono fondi sufficienti per eseguire l' operazione di traderichiesta.
OrderSend	Invia richieste di trade ad un server
OrderSendAsync	Invia in modo asincrono richieste di trade senza attendere la risposta del trade, dal trade server
PositionsTotal	Restituisce il numero di posizioni aperte
PositionGetSymbol	Restituisce il simbolo corrispondente alla posizione aperta
PositionSelect	Sceglie una posizione aperta per un' ulteriore lavorazione con essa
PositionSelectByTicket	Selects a position to work with by the ticket number specified in it
PositionGetDouble	Restituisce la proprietà richiesta di una posizione aperta (double)

Funzione	Azione
PositionGetInteger	Restituisce la proprietà richiesta di una posizione aperta (datetime o int)
PositionGetString	Restituisce la proprietà richiesta di una posizione aperta (string)
PositionGetTicket	Returns the ticket of the position with the specified index in the list of open positions
OrdersTotal	Restituisce il numero di ordini
OrderGetTicket	Restituisce il ticket di un ordine corrispondente
OrderSelect	Seleziona un ordine per un'ulteriore lavoro con esso
OrderGetDouble	Restituisce la proprietà richiesta dell'ordine (double)
OrderGetInteger	Restituisce la proprietà richiesta dell'ordine (datetime o int)
OrderGetString	Restituisce la proprietà richiesta dell'ordine (string)
HistorySelect	Recupera la cronistoria delle transazioni e degli ordini per il periodo di tempo specificato del server time
HistorySelectByPosition	Richiede la cronistoria delle offerte con un determinato position identifier.
HistoryOrderSelect	Seleziona un ordine nella cronistoria per un ulteriore lavoro con esso
HistoryOrdersTotal	Restituisce il numero di ordini nella cronistoria
HistoryOrderGetTicket	Restituisce il ticket dell'ordine corrispondente all' ordine nella cronistoria
HistoryOrderGetDouble	Restituisce la proprietà richiesta di un ordine nella cronistoria (doppia)
HistoryOrderGetInteger	Restituisce la proprietà richiesta di un ordine nella cronistoria (datetime o int)
HistoryOrderGetString	Restituisce la proprietà richiesta di un ordine nella cronistoria (stringa)
HistoryDealSelect	Seleziona un affare nella cronistoria per l'ulteriore chiama attraverso funzioni appropriate
HistoryDealsTotal	Restituisce il numero degli affari nella cronistoria
HistoryDealGetTicket	Restituisce un ticket di un affare corrispondente nella cronistoria
HistoryDealGetDouble	Restituisce la proprietà richiesta di un addare nella cronistoria (double)
HistoryDealGetInteger	Restituisce la proprietà richiesta di un affare nella cronistoria (datetime o int)
HistoryDealGetString	Restituisce la proprietà richiesta di un addare nella cronistoria (string)

OrderCalcMargin

La funzione calcola il margine richiesto per il tipo di ordine specificato, sul corrente account, nel corrente contesto di mercato che non tiene conto dei correnti ordini pendenti e posizioni aperte. Permette la valutazione del margine per l'operazione di trade prevista. Il valore viene restituito nella valuta del conto.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,           // tipo di ordine
   string                symbol,           // nome del simbolo
   double                volume,           // volume
   double                price,            // prezzo di apertura
   double&               margin            // valore per ottenere il valore del margi
   );
```

Parametri

azione

[in] Il tipo di ordine, può essere uno dei valori dell'enumerazione ENUM_ORDER_TYPE.

symbol

[in] Nome del Simbolo.

volume

[in] volume delle operazioni di trade.

price

[in] Prezzo di apertura.

margin

[out] La variabile, per cui il valore del margine richiesto sarà scritto nel caso la funzione viene eseguita correttamente. Il calcolo viene eseguito come se non c'erano ordini pendenti e posizioni aperte, sul corrente account. Il valore del margine dipende da molti fattori, e può variare in diversi contesti di mercato.

Valore restituito

La funzione restituisce true in caso di successo, altrimenti restituisce false. Al fine di ottenere informazioni sull' errore, chiamare la funzione GetLastError().

Vedi anche

OrderSend(), Proprietà degli Ordini, Tipi di Operazioni di Trade

OrderCalcProfit

La funzione calcola il profitto per il corrente account, nelle attuali condizioni di mercato, in base ai parametri passati. La funzione viene utilizzata per la pre-valutazione del risultato di un'operazione di trade. Il valore viene restituito nella valuta del conto.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,          // tipo di ordine (ORDER_TYPE_BUY or ORDER_
   string                symbol,          // nome del simbolo
   double                volume,          // volume
   double                price_open,      // prezzo di apertura
   double                price_close,     // prezzo di chiusura
   double&               profit           // variabile per ottenere il valore del pr
   );
```

Parametri

azione

[in] Tipo di ordine, può essere uno dei due valori dell'enumarazione ENUM_ORDER_TYPE: ORDER_TYPE_BUY o ORDER_TYPE_SELL.

symbol

[in] Nome del Simbolo.

volume

[in] volume delle operazioni di trade.

price_open

[in] Prezzo di apertura.

price_close

[in] Prezzo di chiusura.

profit

[out] La variabile, per cui il valore del profitto calcolato sarà scritto nel caso la funzione viene eseguita correttamente. Il valore di profitto stimato dipende da molti fattori, e può variare nei diversi contesti di mercato.

Valore restituito

La funzione restituisce true in caso di successo, altrimenti restituisce false. Se un tipo di ordine non valido viene specificato, la funzione restituisce false. Al fine di ottenere informazioni sull' errore, chiamare GetLastError().

Vedi anche

OrderSend(), Proprietà degli Ordini, Tipi di Operazioni di Trade

OrderCheck

La funzione OrderCheck() controlla se ci sono abbastanza soldi per eseguire un' operazione di traderichiesta. I risultati del controllo sono posti ai campi della struttura MqlTradeCheckResult.

```
bool  OrderCheck(
   MqlTradeRequest&       request,      // struttura della richiesta
   MqlTradeCheckResult&   result        // struttura del risultato
   );
```

Parametri

request

[in] Puntatore alla struttura del tipo MqlTradeRequest, che descrive l'azione di trade richiesta.

result

[in,out] Puntatore alla struttura del tipo MqlTradeCheckResult, un cui verrà posizionato il risultato del controllo.

Valore restituito

Se i fondi non sono sufficienti per l'operazione, oi parametri sono compilati in modo errato, la funzione restituisce il valore false. In caso di un controllo di base di successo delle strutture (controllo di puntatori), restituisce true. **Tuttavia, questo non significa che l'operazione di trade richiesta, è sicuro che venga eseguita con successo**. Per una descrizione più dettagliata del risultato dell'esecuzione della funzione, analizzare i campi della struttura *result* .

Al fine di ottenere informazioni sull' errore, chiamare la funzione GetLastError().

Vedi anche

OrderSend(), Tipi di Operazione di Trade, Struttura delle Richieste di Trade, Struttura dei Risultati del Controllo della Richiesta, Struttura del Risultato della Richiesta di Trade

OrderSend

OrderSend() viene utilizzato per l'esecuzione di operazioni di trade inviando richieste al trade server .

```
bool  OrderSend(
   MqlTradeRequest&  request,      // struttura della query
   MqlTradeResult&   result        // struttura della risposta
   );
```

Parametri

request

[in] Puntatore ad una struttura di tipo MqlTradeRequest descrivente l'attività di trade del client.

result

[in, out] Puntatore ad una struttura di tipo MqlTradeResult che descrive il risultato dell' operazione di trade in caso di completamento (se viene restituito true).

Valore restituito

In caso di un controllo di base di successo delle strutture (controllo index) restituisce true. **Tuttavia, questo non è segno di esecuzione di un'operazione di trade avvenuta con successo.** Per una descrizione più dettagliata del risultato dell'esecuzione della funzione, analizzare i campi della struttura *result* .

Nota

Le richieste di trade passano attraverso diverse fasi di verifica su un trade server. Prima di tutto, controlla se tutti i campi richiesti del parametro *request* vengono compilati correttamente. Se non ci sono errori, il server accetta l'ordine per ulteriori elaborazioni. Se l'ordine è correttamente accettato dal trade server, la funzione OrderSend() restituisce true.

Si consiglia di verificare la richiesta prima di inviarla ad un un trade server. Per controllare le richieste, utilizzare la funzione OrderCheck(). Essa verifica se ci sono fondi sufficienti per eseguire l'operazione di trade, e restituisce una serie di parametri utili ai risultati dei controlli della richiesta di trade:

- codice di ritorno contenente informazioni sugli errori nella richiesta controllata;
- valore di bilancio che verrà visualizzato dopo che l'operazione di trade è eseguita;
- valore dell'equità che verrà visualizzato dopo che l'operazione di trade è eseguita;
- valore a virgola mobile che verrà visualizzato dopo che l'operazione di trade è eseguita;
- margine richiesto per l'operazione di trade;
- quantità di equità libera che rimarrà dopo l'esecuzione dell' operazione di trade;
- il livello di margine che sarà fissato dopo che l'operazione di trade è eseguita;
- commento al codice di risposta, descrizione dell'errore.

Quando si invia un ordine di mercato (MqlTradeRequest.action=TRADE_ACTION_DEAL), il risultato con successo della funzione OrderSend() non significa che l'ordine sia stato eseguito (i trades appropriati sono stati eseguiti). In questo caso, 'true' significa solo che l'ordine è stato inserito con successo nel sistema di trading per l'ulteriore esecuzione. Il trade server può compilare l' *affare* o *ordine* (valori dei campi) nel risultato della struttura restituito, se è a conoscenza di questi dati quando si forma una risposta ad una chiamata OrderSend(). Generalmente, gli eventi di esecuzione di trades corrispondenti ad un ordine possono verificarsi dopo aver inviato una risposta alla

chiamata OrderSend(). Pertanto, per qualsiasi tipo di richiesta di trade, quando si riceve il risultato delll'esecuzione di OrderSend(), dovremmo prima controllare il codice di responso del trade server *retcode* e (se necessario) il codice di responso del sistema esterno *retcode_external* disponibile nel risultato della struttura ottenuto.

Ogni ordine accettato viene memorizzato sul trade server in attesa di elaborazione fino a quando si verifica una delle condizioni per la sua esecuzione:

- espirazione (scadenza),
- comparsa di una richiesta opposta,
- esecuzione dell'ordine quando appare il prezzo di esecuzione,
- una richiesta di annullare viene ricevuta.

Al momento dell'elaborazione dell'ordine, il trade server invia al terminale un messaggio circa il verificarsi dell'evento di Trade, che può essere elaborato dalla funzione OnTrade().

Il risultato della esecuzione della richiesta di trade su un server, inviata da OrderSend() può essere monitorata dall'handler OnTradeTransaction. Va notato che l'handler OnTradeTransaction verrà chiamato più volte durante l'esecuzione di una richiesta di trade.

Per esempio, quando si invia un ordine di acquisto di mercato, esso viene gestito, un ordine di acquisto appropriata viene creato per l'account, l'ordine quindi viene eseguito e rimosso dalla lista di quelli aperti, e poi viene aggiunto alla cronistoria ordini, un appropriato aff viene aggiunto alla cronistoria ed una nuova posizione viene creata. La funzione OnTradeTransaction verrà chiamata per ognuno di questi eventi.

Esempio:

```
//--- valore per ORDER_MAGIC
input long order_magic=55555;
//+------------------------------------------------------------------------------+
//| Funzione di start del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- si assicura che l'account sia demo
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("L'operazione dello Script non è consentita su un account live!");
      return;
     }
//--- piazza l'ordine o lo cancella
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- nessun ordine correntemente - piazza un ordine
      uint res=SendRandomPendingOrder(order_magic);
      Print("Codice di ritorno del trade server",res);
     }
   else // ci sono ordini - elimina ordini
     {
      DeleteAllOrdersByMagic(order_magic);
     }
```

```
//---
  }
//+------------------------------------------------------------------------------------+
//| Riceve il numero attuale di ordini con ORDER_MAGIC specificato                     |
//+------------------------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- passa attraverso tutti gli ordini pendenti
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------------------------+
//| Elimina tutti gli ordini pendenti con ORDER_MAGIC specificato                      |
//+------------------------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- passa attraverso tutti gli ordini pendenti
   for(int i=OrdersTotal()-1;i>=0;i--)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- ordine con l'appropriato ORDER_MAGIC
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={};
            MqlTradeRequest request={};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- scrive la risposta del server nel registro
            Print(__FUNCTION__,": ",result.comment," reply code ",result.retcode);
           }
//---
  }
//+------------------------------------------------------------------------------------+
//| Importa un ordine pendente in modo casuale                                         |
//+------------------------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- prepara una richiesta
   MqlTradeRequest request={};
   request.action=TRADE_ACTION_PENDING;          // imposta un ordine pendente
   request.magic=magic_number;                   // ORDER_MAGIC
   request.symbol=_Symbol;                       // symbol
   request.volume=0.1;                           // volume in 0.1 lotti
```

```
   request.sl=0;                                  // Stop Loss non è specificato
   request.tp=0;                                  // Take Profit non è specificato
//--- forma il tipo di ordine
   request.type=GetRandomType();                  // tipo di ordine
//--- forma il prezzo per l'ordine pendente
   request.price=GetRandomPrice(request.type);  // open price
//--- invia una richiesta di trade
   MqlTradeResult result={};
   OrderSend(request,result);
//--- scrive la risposta del server nel registro
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- restitiusce il codice della risposta data dal trade server
   return result.retcode;
  }
//+------------------------------------------------------------------------------+
//| Restituisce il tipo di ordine pendente in modo casuale                       |
//+------------------------------------------------------------------------------+
ENUM_ORDER_TYPE GetRandomType()
  {
   int t=MathRand()%4;
//---   0<=t<4
   switch(t)
     {
      case(0):return(ORDER_TYPE_BUY_LIMIT);
      case(1):return(ORDER_TYPE_SELL_LIMIT);
      case(2):return(ORDER_TYPE_BUY_STOP);
      case(3):return(ORDER_TYPE_SELL_STOP);
     }
//--- valore non corretto
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------------------+
//| Restituisce il prezzo in modo casuale                                        |
//+------------------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- livelli di stop per il simbolo
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- riceve i dati dell'ultimo tick
   MqlTick last_tick={};
   SymbolInfoTick(_Symbol,last_tick);
//--- calcola il prezzo in base al tipo
   double price;
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT or ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // partenza dal prezzo Bid
      price=price-(distance+(MathRand()%10)*5)*_Point;
```

```
      }
    else             // ORDER_TYPE_SELL_LIMIT or ORDER_TYPE_BUY_STOP
      {
       price=last_tick.ask; // partenza dal prezzo Ask
       price=price+(distance+(MathRand()%10)*5)*_Point;
      }
//---
    return(price);
  }
```

Vedi anche

Tipi di Operazioni di Trade, Struttura Richieste di Trade, Struttura dei Risultati della Richiesta del Controllo, Struttura del Risultato della Richiesta di Trade

OrderSendAsync

Il OrderSendAsync() viene utilizzato per lo svolgimento di operazioni di trade asincrone, senza attendere la risposta del trade server ad una richiesta inviata. La funzione è progettata per il trading ad alta-frequenza, quando sotto i termini della algoritmo di trading, è inaccettabile perdere tempo in attesa di una risposta dal server.

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // Struttura dela Richiesta
   MqlTradeResult&   result        // Struttura della Risposta
   );
```

Parametri

request

[in] Puntatore alla struttura di tipo MqlTradeRequest che descrive l'azione di trade del client.

result

[in,out] Un puntatore alla struttura di tipo MqlTradeResult che descrive il risultato di un'operazione di trade in caso di corretta esecuzione della funzione (se true viene restituito).

Valore restituito

Restituisce true se la richiesta viene inviata al trade server. Nel caso in cui la richiesta non venga inviata, restituisce false. Nel caso in cui la richiesta venga inviata, nella variabile *result* il codice di risposta contiene il valore TRADE_RETCODE_PLACED (codice 10008) - "ordine piazzato". L' esecuzione di successo significa solo il fatto di inviare, ma non fornisce alcuna garanzia che la richiesta ha raggiunto il trade server ed che sia stata accettata per l'elaborazione. Quando si elabora la richiesta ricevuta, il trade server invia una risposta al terminale client notificando sul cambiamento dello stato attuale delle posizioni, ordini ed affari, che portano alla generazione dell' evento Trade.

Il risultato dell' esecuzione della richiesta di trade sul server inviata da OrderSendAsync() può essere monitorato dall'handler OnTradeTransaction. Va notato che l'handler OnTradeTransaction verrà chiamato più volte durante l'esecuzione di una richiesta di trade.

Per esempio, quando si invia un ordine di acquisto di mercato, esso viene gestito, un ordine di acquisto appropriata viene creato per l'account, l'ordine quindi viene eseguito e rimosso dalla lista di quelli aperti, e poi viene aggiunto alla cronistoria ordini, un appropriato aff viene aggiunto alla cronistoria ed una nuova posizione viene creata. La funzione OnTradeTransaction verrà chiamata per ognuno di questi eventi. Per ottenere tali dati, devono essere analizzati i parametri della funzione:

- **trans** - questo parametro riceve la struttura MqlTradeTransaction che descrive una transazione di trade applicata ad un account;
- **request** - questo parametro riceve la struttura MqlTradeRequest che descrive la richiesta di trade risultata in una transazione di trade;
- **result** - questo parametro riceve la struttura MqlTradeResult che descrive un risultato di esecuzione di una richiesta di trade.

Nota

In termini di obiettivi e parametri, la funzione è simile ad OrderSend(), ma diversamente da essa, è asincrona, cioè non mantiele il funzionamento del programma in attesa del risultato dell'esecuzione

della funzione. È possibile confrontare il tasso delle operazioni di trade di queste due funzioni utilizzando l'Expert Advisor di esempio.

Esempio:

```
#property description "Expert Advisor per inviare richieste di trade "
                      " usando la duznione OrderSendAsync().\r\n"
#property description "Handling degli eventi di trading usando"
                      " le funzioni handler OnTrade() ed OnTradeTransaction() (viene v
#property description "I parametri dell' Expert Advisor consentono di impostare il Mag
                      " (ID univoco) "
#property description "e la modalità di visualizzazione dei messaggi nell' Experts log
//--- parametri di input
input int  MagicNumber=1234567;      // ID Expert Advisor
input bool DescriptionModeFull=true; // Modalità dettagliata di output
//--- variabile per usare nella chiamata HistorySelect()
datetime history_start;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- controlla se l'autotrading è consentito
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("L' Autotrading nel terminale è disabilitato, l' Expert Advisor verrà rimo
      ExpertRemove();
      return(-1);
     }
//--- Non in grado di operare su un conto reale
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("L' Expert Advisor non può fare trading sun un account reale!");
      ExpertRemove();
      return(-2);
     }
//--- verifica se è possibile fare trading su questo account (per esempio, il trading
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Il trading su questo account è disabilitato");
      ExpertRemove();
      return(-3);
     }
//--- salva l'orario di lancio dell'Expert Advisor per la ricezione della cronistoria
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
```

```
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                                           |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- elimina tutti gli oggetti grafici
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------------------+
//| Funzione TradeTransaction                                                    |
//+------------------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- intestazione nominata dopo l'event handler della funzione di trading
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- riceve il tipo di transazione come valore dell'enumerazione
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- se la trascrizione è il risultato di una richiesta di handling
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //--- mostra il nome della transazione
      Print(EnumToString(type));
      //--- quindi mostra la descrizione della stringa della richiesta di handling
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- e mostra la desceizione del risultato della richiesta
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // mostra la descrizione completa della transazione per la transazione di alt
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Funzione di Trade                                                            |
//+------------------------------------------------------------------------------+
void OnTrade()
  {
//--- membro statico per la memorizzazione dello status dell'account di trading
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- richiesta cronistoria di trading
   bool update=HistorySelect(history_start,TimeCurrent());
```

```
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update);
//--- intestazione nominata dopo l'event handler della funzione di trading
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- mostra il nome dell handler ed il numero di ordini al momento dell'handling
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//--- mostra il numero di ordini, posizioni, affari, così come cambi nelle parentesi
   PrintFormat("PositionsTotal() = %d (%+d)",
               curr_positions,(curr_positions-prev_positions));
   PrintFormat("OrdersTotal() = %d (%+d)",
               curr_orders,curr_orders-prev_orders);
   PrintFormat("HistoryOrdersTotal() = %d (%+d)",
               curr_deals,curr_deals-prev_deals);
   PrintFormat("HistoryDealsTotal() = %d (%+d)",
               curr_history_orders,curr_history_orders-prev_history_orders);
//--- inserisce il break di stringa per visualizzare il log più convenientemente
   Print("");
//--- salva lo status dell' account
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------------------+
//| Funzione ChartEvent                                                          |
//+------------------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- handling CHARTEVENT_CLICK event ("Click sul chart")
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- minimo volume per un affare
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- se viene premuto il bottone "Buy", allora compra
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- rilascia il bottone
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
```

```
      //--- se viene premuto il bottone "Sell", allora vende
      if(sparam=="Sell")
        {
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- rilascia il bottone
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Restituisce la descrizione testuale di una transazione                       |
//+------------------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- prepara la stringa per la restituzione dalla funzione
   string desc=EnumToString(trans.type)+"\r\n";
//--- tutti i possibili dati vengono aggiunti in modalità dettagliata
   if(detailed)
     {
      desc+="Simbolo: "+trans.symbol+"\r\n";
      desc+="Ticket dell'affare: "+(string)trans.deal+"\r\n";
      desc+="Tipo dell'Affare: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Ticket dell'Ordine: "+(string)trans.order+"\r\n";
      desc+="Tipo dell'Ordine: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Status dell'Ordine: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Tipo di orario dell'Ordine: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Espirazione(scadenza) dell'Ordine: "+TimeToString(trans.time_expiration)+
      desc+="Prezzo: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Innesco del Prezzo: "+StringFormat("%G",trans.price_trigger)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
      desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- restituisce una stringa ricevuta
   return desc;
  }
//+------------------------------------------------------------------------------+
//| Restituisce la descrizione testuale della richiesta di trade                 |
//+------------------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- prepara la stringa per la restituzione dalla funzione
   string desc=EnumToString(request.action)+"\r\n";
//--- aggiunge tutti i dati disponibili in modalità dettagliata
```

```
   if(detailed)
     {
      desc+="Simbolo: "+request.symbol+"\r\n";
      desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Ticket dell'Ordine: "+(string)request.order+"\r\n";
      desc+="Tipo di Ordine: "+EnumToString(request.type)+"\r\n";
      desc+="Riempimento dell'Ordine: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Tipo di orario dell'Ordine: "+EnumToString(request.type_time)+"\r\n";
      desc+="Espirazione dell'Ordine: "+TimeToString(request.expiration)+"\r\n";
      desc+="Prezzo: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Punti di Deviazione: "+StringFormat("%G",request.deviation)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
      desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
      desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Commento: "+request.comment+"\r\n";
     }
//--- restituisce la stringa ricevuta
   return desc;
  }
//+------------------------------------------------------------------------------------+
//| Restituisce la descrizione testuale del risultato della richiesta di handling  |
//+------------------------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- prepara la stringa per il ritorno dalla funzione
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- aggiunge tutti i dati disponibili in modalità dettagliata
   if(detailed)
     {
      desc+="ID della Richiesta: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Ticket dell'Ordine: "+(string)result.order+"\r\n";
      desc+="Ticket dell'Affare: "+(string)result.deal+"\r\n";
      desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Prezzo: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Commento: "+result.comment+"\r\n";
     }
//--- restituisce la stringa ricevuta
   return desc;
  }
//+------------------------------------------------------------------------------------+
//| Crea due bottoni per l'acquisto e la vendita                                       |
//+------------------------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//--- controlla l'oggetto nominato "Buy"
```

```
   if(ObjectFind(0,"Buy")>=0)
     {
      //--- se l'oggetto trovato non è un bottone, lo elimina
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Buy");
     }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // crea il bottone "Buy"
//--- configura il bottone "Sell"
   ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
   ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
   ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//--- controlla la presenza di un oggetto chiamato "Sell"
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- se l'oggetto trovato non è un bottone, lo elimina
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // crea il bottone "Sell"
//--- configura il bottone "Sell"
   ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
   ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- esegue l'aggiornamento forzato del chart per visualizzare i bottoni immediatamer
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------------------------+
//| Compra usando la funzione asincrona OrderSendAsync()                               |
//+------------------------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- prepara la richiesta
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
```

```
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Buy(compra), usando OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Vende usando la funzione asincrona OrderSendAsync()                          |
//+------------------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- prepara la richiesta
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_SELL;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_BID);
   req.deviation   =10;
   req.comment     ="Sell(vende) usando OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------------------+
```

Esempio di visualizzazione dei messaggi nel registro(Log) "Experts":

```
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnChartEvent: sparam = Sell
 12:52:52   ExpertAdvisor (EURUSD,H1)   Sell EURUSD 0.01 lot
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_REQUEST
 12:52:52   ExpertAdvisor (EURUSD,H1)   ------------RequestDescription
 12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_ACTION_DEAL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Simbolo: EURUSD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number: 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket dell'Ordine: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo dell'Ordine: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Riempimento dell'Ordine: ORDER_FILLING_FOK
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo di orario dell'Ordine: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Espirazione(scadenza) dell'Ordine: 1970.01.01 (
```

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Prezzo: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Punti di Deviazione: 10
 12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Limit: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
 12:52:52   ExpertAdvisor (EURUSD,H1)   Commento: Sell(vende) usando OrderSendAsync()
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ ResultDescription
 12:52:52   ExpertAdvisor (EURUSD,H1)   Retcode 10009
 12:52:52   ExpertAdvisor (EURUSD,H1)   Request ID: 2
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket dell'Ordine: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
 12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Prezzo: 1.29313
 12:52:52   ExpertAdvisor (EURUSD,H1)   Ask: 1.29319
 12:52:52   ExpertAdvisor (EURUSD,H1)   Bid: 1.29313
 12:52:52   ExpertAdvisor (EURUSD,H1)   Commento:
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+2)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+2)
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
 12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_ADD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Simbolo: EURUSD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket dell'Ordine: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo dell'Ordine: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo di orario dell'Ordine: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Espirazione(scadenza) dell'Ordine: 1970.01.01 (
12:52:52   ExpertAdvisor (EURUSD,H1)   Prezzo: 1.29313
 12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
 12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_DELETE
 12:52:52   ExpertAdvisor (EURUSD,H1)   Simbolo: EURUSD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
```

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket dell'Ordine: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo dell'Ordine: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo di orario dell'Ordine: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Espirazione(scadenza) dell'Ordine: 1970.01.01 (
12:52:52   ExpertAdvisor (EURUSD,H1)   Prezzo: 1.29313
 12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
 12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_HISTORY_ADD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Simbolo: EURUSD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket dell'Ordine: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo dell'Ordine: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Status dell'Ordine: ORDER_STATE_FILLED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo di orario dell'Ordine: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Espirazione(scadenza) dell'Ordine: 1970.01.01 (
12:52:52   ExpertAdvisor (EURUSD,H1)   Prezzo: 1.29313
 12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
 12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_DEAL_ADD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Simbolo: EURUSD
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
 12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Ticket dell'Ordine: 16361998
```

```
 12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_BUY
 12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Tipo di orario dell'Ordine: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Espirazione(scadenza) dell'Ordine: 1970.01.01 (
12:52:52   ExpertAdvisor (EURUSD,H1)   Prezzo: 1.29313
 12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
 12:52:52   ExpertAdvisor (EURUSD,H1)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
 12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
 12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
 12:52:52   ExpertAdvisor (EURUSD,H1)
```

PositionsTotal

Restituisce il numero di posizioni aperte.

```
int  PositionsTotal();
```

Valore restituito

Value of int type.

Nota

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Vedi anche

PositionGetSymbol(), PositionSelect(), Proprietà Posizioni

PositionGetSymbol

Restituisce il simbolo corrispondente alla posizione aperta e seleziona automaticamente la posizione per l'ulteriore lavoro con essa utilizzando le funzioni PositionGetDouble, PositionGetInteger, PositionGetString.

```
string  PositionGetSymbol(
   int  index      // Numero nella lista delle posizioni
   );
```

Parametri

index

[in] Numero della posizione nella lista delle posizioni aperte.

Valore restituito

Valore di tipo string. Se la posizione non è stata trovata, verrà restituita una stringa vuota. Per ottenere il codice di errore, chiamare la funzione GetLastError().

Nota

Per ciascun simbolo, in un dato momento di tempo, solo una posizione può essere aperta, che è il risultato di uno o più affari. Da non confondere i correnti ordini pendenti con le posizioni, che sono anch'esse visualizzate nella scheda "Trade" del "BoxAttrezzi" del terminale client.

Il numero totale di posizioni su un trade account non può superare il totale del numero di strumenti finanziari.

Vedi anche

PositionsTotal(), PositionSelect(), Proprietà delle Posizioni

PositionSelect

Sceglie una posizione aperta per l'ulteriore lavoro con essa. Restituisce true se la funzione è completata con successo. Restituisce false in caso di fallimento. Per ottenere informazioni sull'errore, chiamare GetLastError().

```
bool  PositionSelect(
   string  symbol      // Nome del simbolo
   );
```

Parametri

symbol

[in] Nome dello strumento finanziario.

Valore restituito

Valore di tipo bool.

Nota

Per ciascun simbolo, in un dato momento di tempo, solo una posizione può essere aperta, che è il risultato di uno o più affari. Da non confondere posizioni con correnti ordini pendenti, che sono anche visualizzate nella scheda "Trade" del "BoxAttrezzi" del terminale client.

La funzione PositionSelect() copia i dati relativi ad una posizione nell'ambiente del programma, e successive chiamate di PositionGetDouble(), PositionGetInteger() e PositionGetString() restituiscono i dati precedentemente copiati. Questo significa che la posizione stessa non esiste più (o il suo volume, direzione, ecc. è cambiato), ma i dati di quela posizione possono ancora essere ottenuti. Per garantire la ricezione di nuovi dati su una posizione, si consiglia di chiamare PositionSelect() giusto prima di fare riferimento ad essi.

Vedi anche

PositionGetSymbol(), PositionsTotal(), Proprietà della Posizione

PositionSelectByTicket

Selects an open position to work with based on the ticket number specified in the position. If successful, returns true. Returns false if the function failed. Call GetLastError() for error details.

```
bool  PositionSelectByTicket(
   ulong   ticket     // Position ticket
   );
```

Parameters

ticket

[in] Position ticket.

Return Value

A value of the bool type.

Note

The PositionSelectByTicket() function copies position data to the program environment. Further calls of PositionGetDouble(), PositionGetInteger() and PositionGetString() return the previously copied data. Even if a position does not exist already (or its size, direction etc. has changed), the data may still be received sometimes. To make sure that you receive valid position data, it is recommended to call PositionSelectByTicket() before you access the data.

See also

PositionGetSymbol(), PositionsTotal(), Position Properties

PositionGetDouble

La funzione restituisce la proprietà richiesta di una posizione aperta, pre-selezionata usando PositionGetSymbol o PositionSelect. La proprietà della funzione deve essere di tipo double. Ci sono 2 varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // Property identifier
   );
```

2. Restituisce true o false, a seconda del successo della esecuzione della funzione. If successful, the value of the property is placed in a receiving variable passed by reference by the last parameter.

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,     // Property identifier
   double&                        double_var       // Qui si accetta il valore della p
   );
```

Parametri

property_id

[in] Identificatore della proprietà della posizione. Il valore può essere uno dei valori dell'enumerazione ENUM_POSITION_PROPERTY_DOUBLE.

double_var

[out] Variabile di tipo double, che accetta il valore della proprietà richiesta.

Valore restituito

Valore del tipo double. Se la funzione fallisce, viene restituito 0.

Nota

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Per garantire la ricezione dei nuovi dati sulla posizione, si raccomanda di chiamare PositionSelect() giusto prima di fare riferimento ad essi.

Vedi anche

PositionGetSymbol(), PositionSelect(), Proprietà Posizioni

PositionGetInteger

La funzione restituisce la proprietà richiesta di una posizione aperta, pre-selezionata usando PositionGetSymbol o PositionSelect. La proprietà della posizione deve essere datetime, tipo int. Ci sono 2 varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
long PositionGetInteger(
ENUM_POSITION_PROPERTY_INTEGERPROPERTY_ID// Property identifier
   );
```

2. Restituisce true o false, a seconda del successo della esecuzione della funzione. In caso di successo, il valore della proprietà è posto in una variabile passata per riferimento per l'ultimo parametro.

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // Property identifier
   long&                           long_var         // Qui si accettano i valori della
   );
```

Parametri

property_id

[in] Identificatore della proprietà della posizione. Il valore può essere uno dei valori dell'enumerazione ENUM_POSITION_PROPERTY_INTEGER.

long_var

[out] Variabile di tipo long che accetta il valore della proprietà richiesta.

Valore restituito

Valore del tipo long. Se la funzione fallisce, viene restituito 0.

Nota

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Per garantire la ricezione dei nuovi dati sulla posizione, si raccomanda di chiamare PositionSelect() giusto prima di fare riferimento ad essi.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione di Trade                                                |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- controlla se una posizione è presente e visualizza l'orario del suo cambiamento
```

```
   if(PositionSelect(_Symbol))
     {
//--- riceve l'ID della posizione per l'ulteriore lavoro
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," posizione #",position_ID);
//--- riceve l'orario di formazione della posizione in millisecondi dal 01.01.1970
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Posizione #%d  POSITION_TIME_MSC = %i64 millisecondi => %s",positic
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- riceve l'orario dell'ultimo cambiamento della posizione, in secondi, dal 01.01.1
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE = %i64 secondi => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- riceve l'orario dell'ultimo cambiamento della posizione, in millisecondi, dal 01
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE_MSC = %i64 millisecondi => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

Vedi anche

PositionGetSymbol(), PositionSelect(), Proprietà Posizioni

PositionGetString

La funzione restituisce la proprietà richiesta di una posizione aperta, pre-selezionata usando PositionGetSymbol o PositionSelect. La proprietà della posizione deve essere di tipo stringa. Ci sono 2 varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // Property identifier
   );
```

2. Restituisce true o false, a seconda del successo della esecuzione della funzione. In caso di successo, il valore della proprietà è posto in una variabile passata per riferimento per l'ultimo parametro.

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // Property identifier
   string&                        string_var       // Qui si accetta il valore della p
   );
```

Parametri

property_id

[in] Identificatore della proprietà della posizione. Il valore può essere uno dei valori dell'enumerazione ENUM_POSITION_PROPERTY_STRING.

string_var

[out] Variabile di tipo stringa che accetta il valore della proprietà richiesta.

Valore restituito

Valore di tipo string. Se la funzione fallisce, viene restituita una stringa vuota.

Nota

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Per garantire la ricezione dei nuovi dati sulla posizione, si raccomanda di chiamare PositionSelect() giusto prima di fare riferimento ad essi.

Vedi anche

PositionGetSymbol(), PositionSelect(), Proprietà Posizioni

PositionGetTicket

The function returns the ticket of a position with the specified index in the list of open positions and automatically selects the position to work with using functions PositionGetDouble, PositionGetInteger, PositionGetString.

```
ulong  PositionGetTicket(
   int  index      // The number of a position in the list
   );
```

Parameters

index

[in] The index of a position in the list of open positions, numeration starts with 0.

Return Value

The ticket of the position. Returns 0 if the function fails.

Note

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

To ensure receipt of fresh data about a position, it is recommended to call PositionSelect() right before referring to them.

See also

PositionGetSymbol(), PositionSelect(), Position Properties

OrdersTotal

Restituisce il numero dei correnti ordini.

```
int  OrdersTotal();
```

Valore restituito

Valore di tipo int.

Nota

Non confondere i correnti ordini pendenti con posizioni, che sono anche visualizzati nella scheda "Trade" del "BoxAttrezzi" del terminale client. Un ordine è una richiesta di condurre un' operazione, mentre una posizione è il risultato di uno o più affari.

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Vedi anche

OrderSelect(), OrderGetTicket(), Proprietà degli Ordini

OrderGetTicket

Restituisce il Ticket di un ordine corrispondente e seleziona automaticamente l'ordine per un ulteriore lavoro con esso utilizzando le funzioni.

```
ulong  OrderGetTicket(
   int  index      // Numero nella lista degli ordini
   );
```

Parametri

index

[in] Numero di un ordine nella lista degli ordini pendenti.

Valore restituito

Valore del tipo ulong . Se la funzione fallisce, viene restituito 0.

Nota

Non confondere i correnti ordini pendenti con posizioni, che sono anche visualizzati nella scheda "Trade" del "BoxAttrezzi" del terminale client. Un ordine è una richiesta di condurre un' operazione, mentre una posizione è il risultato di uno o più affari.

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

La funzione OrderGetTicket() copia i dati di un ordine nell'ambiente del programma, e successive chiamate di OrderGetDouble(), OrderGetInteger(), OrderGetString() restituiscono i dati precedentemente copiati. Ciò significa che l'ordine stesso non esiste più (o il suo prezzo di apertura, Stop Loss/Take Profit o scadenza sono cambiati), ma i dati di questo ordine possono essere ancora ottenuti. Per garantire la ricezione di nuovi dati relativi ad un ordine, si consiglia di chiamare OrderGetTicket() giusto prima di fare riferimento ad essi.

Esempio:

```
voidOnStart()
  {
//---variabili per la restituzione di valori dalle proprietà dell'ordine
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- numero degli attuali ordini in corso
   uint     total=OrdersTotal();
```

```
//--- passa attraverso gli ordini in un ciclo
   for(uint i=0;i<total;i++)
     {
      //--- restituisce il ticket dell'ordine per la sua posizione nella lista
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- restituisce le proprietà dell'ordine
         open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
         time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
         symbol        =OrderGetString(ORDER_SYMBOL);
         order_magic   =OrderGetInteger(ORDER_MAGIC);
         positionID    =OrderGetInteger(ORDER_POSITION_ID);
         initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
         type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
         //--- prepara e mostra le informazioni sull'ordine
         printf("il #ticket %d %s %G %s at %G è stato impostato a %s",
                ticket,                 // ticket ordine
                type,                   // tipo
                initial_volume,         // volume piazzato
                symbol,                 // simbolo
                open_price,             // prezzo di apertura specificato
                TimeToString(time_setup)// tempo di piazzamento dell'ordine
                );
        }
     }
//---
  }
```

Vedi anche

OrdersTotal(), OrderSelect(), OrderGetInteger()

OrderSelect

Seleziona un ordine con cui lavorare. Restituisce true se la funzione è stata completata con successo. Restituisce false se il completamento della funzione non è riuscito. Per ulteriori informazioni su un errore chiamare GetLastError().

```
bool  OrderSelect(
   ulong    ticket      // Ticket ordine
   );
```

Parametri

ticket

[in] Order ticket.

Valore restituito

Valore di tipo bool.

Nota

Non confondere i correnti ordini pendenti con posizioni, che sono anche visualizzati nella scheda "Trade" del "BoxAttrezzi" del terminale client.

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

La funzione OrderSelect() copia i dati di un ordine nell'ambiente del programma, e successive chiamate di OrderGetDouble(), OrderGetInteger(), OrderGetString() restituiscono i dati precedentemente copiati. Ciò significa che l'ordine stesso non esiste più (o il suo prezzo di apertura, Stop Loss/Take Profit o scadenza sono cambiati), ma i dati di questo ordine possono essere ancora ottenuti. Per garantire la ricezione di nuovi dati relativi ad un ordine, si consiglia di chiamare OrderSelect() giusto prima di fare riferimento ad essi.

Vedi anche

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Order Properties

OrderGetDouble

Restituisce la proprietà richiesta di un ordine, pre-selezionato utilizzando OrderGetTicket o OrderSelect. La proprietà dell'ordine deve essere di tipo double. Ci sono 2 varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // Property identifier
   );
```

2. Restituisce true o false, a seconda se la funzione ha avuto successo. If successful, the value of the property is placed in a target variable passed by reference by the last parameter.

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // Property identifier
   double&                     double_var         // Qui si accettano i valori dell
   );
```

Parametri

property_id

[in] Identificatore della proprietà dell'ordine. Il valore può essere uno dei valori dell' enumarazione ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Variabile del tipo double che accetta il valore della proprietà richiesta.

Valore restituito

Valore del tipo double. Se la funzione fallisce, viene restituito 0.

Nota

Non confondere i correnti ordini pendenti con posizioni, che sono anche visualizzati nella scheda "Trade" del "BoxAttrezzi" del terminale client.

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Per garantire la ricezione di dati freschi su un ordine, si consiglia di chiamare OrderSelect() giusto prima di fare riferimento ad essi.

Vedi anche

OrdersTotal(), OrderGetTicket(), Proprietà degli Ordini

OrderGetInteger

Restituisce la proprietà ordine richiesta, pre-selezionto utilizzando OrderGetTicket o OrderSelect. Le proprietà degli ordini devono essere del datetime, tipo int. Ci sono 2 varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // Property identifier
   );
```

2. Restituisce true o false a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // Proprietà Identificatore
   long&                        long_var           // Qui si accetta il valore della p
   );
```

Parametri

property_id

[in] Identificatore della proprietà dell'ordine. Il valore può essere uno dei valori dell' enumerazione ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Variabile di tipo long che accetta il valore della proprietà richiesta.

Valore restituito

Valore del tipo long. Se la funzione fallisce, viene restituito 0.

Nota

Non confondere i correnti ordini pendenti con posizioni, che sono anche visualizzati nella scheda "Trade" del "BoxAttrezzi" del terminale client.

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Per garantire la ricezione di dati freschi su un ordine, si consiglia di chiamare OrderSelect() giusto prima di fare riferimento ad essi.

Vedi anche

OrdersTotal(), OrderGetTicket(), Proprietà degli Ordini

OrderGetString

Restituisce la proprietà ordine richiesta, pre-selezionto utilizzando OrderGetTicket o OrderSelect. La proprietà dell'ordine deve essere di tipo stringa. Ci sono 2 varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // Proprietà Identificatore
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,       // Proprietà Identificatore
   string&                     string_var         // Qui si accetta il valore della pr
   );
```

Parametri

property_id

[in] Identificatore della proprietà dell'ordine. Il valore può essere uno dei valori dell' enumerazione ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Variabile di tipo stringa che accetta il valore della proprietà richiesta.

Valore restituito

Valore di tipo stringa.

Nota

Non confondere i correnti ordini pendenti con posizioni, che sono anche visualizzati nella scheda "Trade" del "BoxAttrezzi" del terminale client.

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Per garantire la ricezione di dati freschi su un ordine, si consiglia di chiamare OrderSelect() giusto prima di fare riferimento ad essi.

Vedi anche

OrdersTotal(), OrderGetTicket(), Proprietà degli Ordini

HistorySelect

Recupera la cronistoria degli affari e degli ordini per il periodo di tempo specificato del server time

```
bool  HistorySelect(
   datetime  from_date,      // Dalla data di
   datetime  to_date         // Alla data di
   );
```

Parametri

from_date

[in] Data di inizio della richiesta.

to_date

[in] Fine della data della richiesta.

Valore restituito

Restituisce true se ha successo, altrimenti restituisce false.

Nota

HistorySelect() crea un elenco di ordini e un elenco di trades in un programma-mql5, per l'ulteriore riferimento agli elementi di lista che usano le funzioni corrispondenti. La grandezza dell'elenco degli affari può essere restituita con la funzione HistoryDealsTotal (), la grandezza della lista degli ordini nella cronistoria può essere ottenuta usando HistoryOrdersTotal(). La selezione nella lista degli ordini dovrebbe essere meglio eseguita da HistoryOrderGetTicket(), per gli elementi nella lista delle offerte HistoryDealGetTicket() si adatta meglio.

Dopo aver usato HistoryOrderSelect(), l'elenco degli ordini nella cronistoria disponibile per il programma mql5 è resettata e nuovamente riempita dall'ordine trovato, se la ricerca di un ordine da parte del ticket è stata completata con successo. Lo stesso vale per l'elenco delle offerte disponibili per il programma mql5 - è resettato da HistoryDealSelect() e riempito nuovamente in caso di avvenuta ricezione di un numero di ticket di un affare.

Esempio:

```
voidOnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- richiede la cronistoria di trade
   HistorySelect(0,TimeCurrent());
//--- crea oggetti
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
```

```
   long     entry;
//--- per tutti gli affari
   for(uint i=0;i<total;i++)
     {
//--- cerca di ottenere i ticket degli affari
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- ottiene le proprietà delle offerte
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
         time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- solo per simbolo corrente
         if(price && time && symbol==Symbol())
           {
            //--- crea l'oggetto prezzo
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- imposta le proprietà dell'oggetto
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit
           }
        }
     }
//--- applica sul grafico
   ChartRedraw();
  }
```

Vedi anche

HistoryOrderSelect(), HistoryDealSelect()

HistorySelectByPosition

Recupera la cronistoria di affari ed ordini con il position identifier specificato.

```
bool  HistorySelectByPosition(
   long   position_id     // position identifier - POSITION_IDENTIFIER
   );
```

Parametri

position_id

[in] Position identifier che è impostato per ogni ordine eseguito ed ogni affare.

Valore restituito

Restituisce true se ha successo, altrimenti restituisce false.

Nota

Non confondere gli ordini della cronistoria di trading con gli attuali ordini pendenti che vengono visualizzati nella scheda "Trade" della barra "BoxAttrezzi". L'elenco degli ordini che sono stati cancellati o hanno portato ad una transazione, possono essere visualizzati nella scheda "Cronistoria" del "BoxAttrezzi" del terminale client.

HistorySelectByPosition() crea in un programma mql5 un elenco di ordini ed un elenco degli affari con un determinato position identifier per ulteriore riferimento agli elementi dell'elenco utilizzando le funzioni appropriate. Per conoscere la grandezza della lista delle offerte, utilizzare la funzione HistoryDealsTotal(), la grandezza della lista degli ordini nella cronistoria può essere ottenuta usando HistoryOrdersTotal(). Per scorrere attraverso gli elementi della lista degli ordini, utilizzare HistoryOrderGetTicket(), per gli elementi della lista delle offerte - HistoryDealGetTicket().

Dopo aver usato HistoryOrderSelect(), l'elenco degli ordini della cronistoria disponibili per il programma mql5 viene azzerato e riempito di nuovo con l'ordine trovato, se la ricerca di un ordine dal suo ticket ha avuto successo. Lo stesso si riferisce alla lista delle offerte disponibili per il programma mql5 - viene resettato dalla funzione HistoryDealSelect() ed è riempito di nuovo, se un affare è stato trovato con successo dal numero di ticket.

Vedi anche

HistorySelect(), HistoryOrderGetTicket(), Proprietà degli Ordini

HistoryOrderSelect

Seleziona un ordine dalla cronistoria per ulteriori chiamate attraverso funzioni appropriate. Restituisce true se la funzione è stata completata con successo. Restituisce false se la funzione ha fallito. Per ulteriori dettagli sull errore chiamare GetLastError().

```
bool  HistoryOrderSelect(
   ulong  ticket      // Ticket Ordine
   );
```

Parametri

ticket

[in] Order ticket.

Valore restituito

Restituisce true se ha successo, altrimenti false.

Nota

Non confondere gli ordini della cronistoria di trading con gli attuali ordini pendenti che vengono visualizzati nella scheda "Trade" della barra "BoxAttrezzi". L'elenco degli ordini che sono stati cancellati o hanno portato ad una transazione, possono essere visualizzati nella scheda "Cronistoria" del "BoxAttrezzi" del terminale client.

HistoryOrderSelect() cancella in un programma-mql5 l'elenco degli ordini dalla cronistoria, disponibile per le chiamate, e copia in esso un singolo ordine, se l'esecuzione di HistoryOrderSelect() è stata completato con successo. Se avete bisogno di passare attraverso tutti gli affari selezionati da HistorySelect(), si dovrebbe meglio utilizzare HistoryOrderGetTicket().

Vedi anche

HistorySelect(), HistoryOrderGetTicket(), Proprietà degli Ordini

HistoryOrdersTotal

Restituisce il numero di ordini nella cronistoria. Prima di chiamare HistoryOrdersTotal(), per prima cosa è necessario ricevere la cronistoria di offerte ed ordini con la funzione HistorySelect() o la funzione HistorySelectByPosition().

```
int  HistoryOrdersTotal();
```

Valore restituito

Valore di tipo int.

Nota

Non confondere gli ordini della cronistoria di trading con gli attuali ordini pendenti che vengono visualizzati nella scheda "Trade" della barra "BoxAttrezzi". L'elenco degli ordini che sono stati cancellati o hanno portato ad una transazione, possono essere visualizzati nella scheda "Cronistoria" del "BoxAttrezzi" del terminale client.

Vedi anche

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Proprietà Ordini

HistoryOrderGetTicket

Restituisce il ticket di un ordine corrispondente nella cronistoria. Prima di chiamare HistoryOrderGetTicket(), per prima cosa è necessario ricevere la storia di offerte ed ordini usando le funzioni HistorySelect() o HistorySelectByPosition().

```
ulong  HistoryOrderGetTicket(
   int  index      // Numero nella lista degli ordini
   );
```

Parametri

index

[in] Numero dell' ordine nella lista degli ordini.

Valore restituito

Valore del tipo ulong . Se la funzione fallisce, viene restituito 0.

Nota

Non confondere gli ordini della cronistoria di trading con gli attuali ordini pendenti che vengono visualizzati nella scheda "Trade" della barra "BoxAttrezzi". L'elenco degli ordini che sono stati cancellati o hanno portato ad una transazione, possono essere visualizzati nella scheda "Cronistoria" del "BoxAttrezzi" del terminale client.

Esempio:

```
voidOnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- richiede l'intera cronistoria
   HistorySelect(from,to);
//---variabili per la restituzione di valori dalle proprietà dell'ordine
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- numero degli attuali ordini in corso
   uint     total=HistoryOrdersTotal();
//--- passa attraverso gli ordini in un ciclo
   for(uint i=0;i<total;i++)
     {
      //--- restituisce il ticket dell'ordine per la sua posizione nella lista
      if((ticket=HistoryOrderGetTicket(i))>0)
        {
```

```
         //--- restituisce le proprietà dell'ordine
         open_price    =HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
         time_setup    =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
         time_done     =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
         symbol        =HistoryOrderGetString(ticket,ORDER_SYMBOL);
         order_magic   =HistoryOrderGetInteger(ticket,ORDER_MAGIC);
         positionID    =HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
         initial_volume=HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
         type          =GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
         //--- prepara e mostra le informazioni sull'ordine
         printf("#ticket %d %s %G %s at %G era impostato a %s => fatto a %s, pos ID=%d
                ticket,                  // ticket dell'ordine
                type,                    // tipo
                initial_volume,          // volume piazzato
                symbol,                  // simbolo
                open_price,              // prezzo open specificato
                TimeToString(time_setup),// orario di piazzamento dell'ordine
                TimeToString(time_done), // orario di esecuzione o cancellazione dell
                positionID               // ID di una posizione, per cui la quantità d
                );
        }
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Restituisce il nome stringa del tipo di ordine                               |
//+------------------------------------------------------------------------------+
string GetOrderType(long type)
  {
   string str_type="unknown operation";
   switch(type)
     {
      case (ORDER_TYPE_BUY):             return("buy");
      case (ORDER_TYPE_SELL):            return("sell");
      case (ORDER_TYPE_BUY_LIMIT):       return("buy limit");
      case (ORDER_TYPE_SELL_LIMIT):      return("sell limit");
      case (ORDER_TYPE_BUY_STOP):        return("buy stop");
      case (ORDER_TYPE_SELL_STOP):       return("sell stop");
      case (ORDER_TYPE_BUY_STOP_LIMIT):  return("buy stop limit");
      case (ORDER_TYPE_SELL_STOP_LIMIT):return("sell stop limit");
     }
   return(str_type);
  }
```

Vedi anche

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Proprietà Ordini

HistoryOrderGetDouble

Restituisce la proprietà richiesta dell'ordine. La proprietà dell'ordine deve essere di tipo double. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
double  HistoryOrderGetDouble(
   ulong                      ticket_number,     // Ticket
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // Property identifier
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  HistoryOrderGetDouble(
   ulong                      ticket_number,     // Ticket
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // Proprietà Identificatore
   double&                    double_var         // Qui si accetta il valore della pr
   );
```

Parametri

ticket_number

[in] Order ticket.

property_id

[in] Identificatore della proprietà dell'ordine. Il valore può essere uno dei valori dell' enumarazione ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Variabile del tipo double che accetta il valore della proprietà richiesta.

Valore restituito

Valore del tipo double.

Nota

Non confondere gli ordini della cronistoria di trading con gli attuali ordini pendenti che vengono visualizzati nella scheda "Trade" della barra "BoxAttrezzi". L'elenco degli ordini che sono stati cancellati o hanno portato ad una transazione, possono essere visualizzati nella scheda "Cronistoria" del "BoxAttrezzi" del terminale client.

Vedi anche

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Proprietà Ordini

HistoryOrderGetInteger

Restituisce la proprietà richiesta di un ordine. La proprietà dell'ordine deve essere datetime, tipo int. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
long  HistoryOrderGetInteger(
   ulong                        ticket_number,     // Ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // Property identifier
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  HistoryOrderGetInteger(
   ulong                        ticket_number,     // Ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // Proprietà Identificatore
   long&                        long_var           // Qui si accetta il valore della p
   );
```

Parametri

ticket_number

[in] Order ticket.

property_id

[in] Identificatore della proprietà dell'ordine. Il valore può essere uno dei valori dell' enumerazione ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Variabile di tipo long che accetta il valore della proprietà richiesta.

Valore restituito

Valore di tipo long.

Nota

Non confondere gli ordini della cronistoria di trading con gli attuali ordini pendenti che vengono visualizzati nella scheda "Trade" della barra "BoxAttrezzi". L'elenco degli ordini che sono stati cancellati o hanno portato ad una transazione, possono essere visualizzati nella scheda "Cronistoria" del "BoxAttrezzi" del terminale client.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Funzione di Trade                                                            |
//+------------------------------------------------------------------------------+
void OnTrade()
  {
//--- riceve i ticket dell'ultimo ordine dalla cronistoria di trading della settimana
   ulong last_order=GetLastOrderTicket();
```

```
   if(HistoryOrderSelect(last_order))
     {
      //--- orario di esecuzione dell'ordine in millisecondi dal 01.01.1970
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Ordine #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- orario di esecuzione/annullamento dell'ordine, in millisecondi dal 01.01.1
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Ordine #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // notifica di fallimento
      PrintFormat("HistoryOrderSelect() fallito per #%d. Codice Errore=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------------------+
//| Restituisce il ticket dell'ultimo ordine nella cronistoria oppure -1         |
//+------------------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//--- richiede la cronistoria per gli ultimi 7 giorni
   if(!GetTradeHistory(7))
     {
      //--- notifica sull'insuccesso della chiamata e restituisce -1
      Print(__FUNCTION__," HistorySelect() ha restituito false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- lavora con gli ordini se ce ne sono
   if(orders>0)
     {
      Print("Ordini = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("primo_ordine = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("ultimo_ordine = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- nessun ordine trovato, restituisco -1
   return -1;
  }
//+------------------------------------------------------------------------------+
```

```
//| Richiede la cronistoria per gli ultimi giorni e restituisce false in caso di falli
//+------------------------------------------------------------------------------------
bool GetTradeHistory(int days)
  {
//--- imposta il periodo di una settimana per richiedere la cronistoria di trading
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//---- fa una richiesta e controlla il risultato
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- cronitoria ricevuta con successo
   return true;
  }
```

Vedi anche

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Proprietà Ordini

HistoryOrderGetString

Restituisce la proprietà richiesta di un ordine. La proprietà dell'ordine deve essere di tipo stringa. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
string  HistoryOrderGetString(
   ulong                      ticket_number,     // Ticket
   ENUM_ORDER_PROPERTY_STRING  property_id       // Proprietà Identificatore
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  HistoryOrderGetString(
   ulong                      ticket_number,     // Ticket
   ENUM_ORDER_PROPERTY_STRING  property_id,       // Proprietà Identificatore
   string&                    string_var         // Qui si accetta il valore della pr
   );
```

Parametri

ticket_number

[in] Order ticket.

property_id

[in] Identificatore della proprietà dell'ordine. Il valore può essere uno dei valori dell' enumerazione ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Variabile di tipo stringa.

Valore restituito

Valore di tipo stringa.

Nota

Non confondere gli ordini della cronistoria di trading con gli attuali ordini pendenti che vengono visualizzati nella scheda "Trade" della barra "BoxAttrezzi". L'elenco degli ordini che sono stati cancellati o hanno portato ad una transazione, possono essere visualizzati nella scheda "Cronistoria" del "BoxAttrezzi" del terminale client.

Vedi anche

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Proprietà Ordini

HistoryDealSelect

Seleziona un affare nella cronistoria per un ulteriore chiamata attraverso funzioni appropriate. Restituisce true se la funzione è stata completata con successo. Restituisce false se la funzione ha fallito. Per ulteriori dettagli sull errore chiamare GetLastError().

```
bool  HistoryDealSelect(
   ulong  ticket      // Ticket dell'affare
   );
```

Parametri

ticket

[in] Deal ticket.

Valore restituito

Restituisce true se ha successo, altrimenti false.

Nota

Attenzione a non confondere ordini, affari e posizioni. Ogni affare è il risultato dell'esecuzione di un ordine, ogni posizione è il risultato di riepilogo di uno o più affari.

HistoryDealSelect() cancella in un programma-mql5 la lista degli affari disponibili per riferimento, e copia il singolo affare, se l'esecuzione di HistoryDealSelect() è stata completata con successo. Se avete bisogno di passare attraverso tutti gli affari selezionati dalla funzioneHistorySelect(), è meglio usare HistoryDealGetTicket().

Vedi anche

HistorySelect(), HistoryDealGetTicket(), Proprietà Affare

HistoryDealsTotal

Restituisce il numero di affare nella cronistoria. Prima di chiamare HistoryDealsTotal(), per prima cosa è necessario ricevere la cronistoria di affari ed ordini con la funzione HistorySelect() o HistorySelectByPosition().

```
int  HistoryDealsTotal();
```

Valore restituito

Valore di tipo int.

Nota

Attenzione a non confondere ordini, affari e posizioni. Ogni affare è il risultato dell'esecuzione di un ordine, ogni posizione è il risultato di riepilogo di uno o più affari.

Vedi anche

HistorySelect(), HistoryDealGetTicket(), Proprietà Affare

HistoryDealGetTicket

La funzione seleziona un accordo per l'ulteriore elaborazione e restituisce il ticket nella cronistoria. Prima di chiamare HistoryDealGetTicket(), per prima cosa è necessario ricevere la cronitoria di affari ed ordini con le funzioni HistorySelect() o HistorySelectByPosition().

```
ulong  HistoryDealGetTicket(
   int  index      // ticket affare
   );
```

Parametri

index

[in] Numero di un affare nella lista degli affari

Valore restituito

Valore del tipo ulong . Se la funzione fallisce, viene restituito 0.

Nota

Attenzione a non confondere ordini, affari e posizioni. Ogni affare è il risultato dell'esecuzione di un ordine, ogni posizione è il risultato di riepilogo di uno o più affari.

Esempio:

```
voidOnStart()
  {
   ulong deal_ticket;            // ticket dell'affare
   ulong order_ticket;           // ticket dell'ordine su cui l'affare è stato eseguit
   datetime transaction_time;    // orario di esecuzione dell'affare
   long deal_type ;              // tipo di operazione di trade
   long position_ID;             // ID della posizione
   string deal_description;      // descrizione dell'operazione
   double volume;                // volume dell'operazione
   string symbol;                // simbolo dell'affare
//--- imposta la data di inizio e di fine per richiedere la cronistoria degli affari
   datetime from_date=0;         // dall'inizio inizio
   datetime to_date=TimeCurrent();// fino al momento corrente
//--- richiede la cronistoria degli affari all'interno del periodo specificato
   HistorySelect(from_date,to_date);
//--- numero totale nella lista degli affari
   int deals=HistoryDealsTotal();
//--- ora elabora ogni trade
   for(int i=0;i<deals;i++)
     {
      deal_ticket=               HistoryDealGetTicket(i);
      volume=                    HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
      transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
      order_ticket=              HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
      deal_type=                 HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
      position_ID=               HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=          GetDealDescription(deal_type,volume,symbol,order_tic
      //--- esegue la buona formattazione per il numero dell'affare
      string print_index=StringFormat("% 3d",i);
      //--- mostra informazioni sull'affare
      Print(print_index+": deal #",deal_ticket," at ",transaction_time,deal_descriptio
     }
  }
//+------------------------------------------------------------------------------+
//| Restituisce la descrizione della stringa dell'operazione                     |
//+------------------------------------------------------------------------------+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
  {
   string descr;
//---
   switch(deal_type)
     {
      case DEAL_TYPE_BALANCE:                  return ("balance");
      case DEAL_TYPE_CREDIT:                   return ("credit");
      case DEAL_TYPE_CHARGE:                   return ("charge");
      case DEAL_TYPE_CORRECTION:               return ("correction");
      case DEAL_TYPE_BUY:                      descr="buy"; break;
      case DEAL_TYPE_SELL:                     descr="sell"; break;
      case DEAL_TYPE_BONUS:                    return ("bonus");
      case DEAL_TYPE_COMMISSION:               return ("additional commission");
      case DEAL_TYPE_COMMISSION_DAILY:         return ("daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY:       return ("monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY:   return ("daily agent commission");
```

HistorySelect(),HistoryDealsTotal(), HistoryDealSelect(), Proprietà Affari

HistoryDealGetDouble

Restituisce la proprietà richiesta di un affare. La proprietà dell' affare deve essere di tipo double. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,     // Ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id        // Identificatore Proprietà identifie
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,     // Ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,       // Identificatore Proprietà
   double&                    double_var         // Qui si accetta il valore della pr
   );
```

Parametri

ticket_number

[in] Deal ticket.

property_id

[in] Identificatore di una proprietà affare. Il valore può essere uno dei valori dell' enumerazione ENUM_DEAL_PROPERTY_DOUBLE.

double_var

[out] Variabile del tipo double che accetta il valore della proprietà richiesta.

Valore restituito

Valore del tipo double.

Nota

Attenzione a non confondere ordini, affari e posizioni. Ogni affare è il risultato dell'esecuzione di un ordine, ogni posizione è il risultato di riepilogo di uno o più affari.

Vedi anche

HistorySelect(), HistoryDealsTotal(), HistoryDealGetTicket(), Deal Properties

HistoryDealGetInteger

Restituisce la proprietà richiesta di un affare. La proprietà dell' affare deve essere datetime, tipo int. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
long  HistoryDealGetInteger(
   ulong                      ticket_number,     // Ticket
   ENUM_DEAL_PROPERTY_INTEGER  property_id       // Proprietà Identificatore
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  HistoryDealGetInteger(
   ulong                      ticket_number,     // Ticket
   ENUM_DEAL_PROPERTY_INTEGER  property_id,       // Proprietà Identificatore identif
   long&                      long_var           // Qui si accetta il valore della p
   );
```

Parametri

ticket_number

[in] Trade ticket.

property_id

[in] Identificatore della proprietà dell' affare. Il valore può essere uno dei valori dell'enumerazione ENUM_DEAL_PROPERTY_INTEGER.

long_var

[out] Variabile di tipo long che accetta il valore della proprietà richiesta.

Valore restituito

Valore di tipo long.

Nota

Attenzione a non confondere ordini, affari e posizioni. Ogni affare è il risultato dell'esecuzione di un ordine, ogni posizione è il risultato di riepilogo di uno o più affari.

Esempio:

```
//+------------------------------------------------------------------------------+
//| Funzione di Trade                                                            |
//+------------------------------------------------------------------------------+
void OnTrade()
  {
//--- riceve i ticket dell'ultimo affare dalla cronistoria di trading della settimana
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
     {
```

```
      //--- orario di esecuzione dell'affare in millisecondi dal 01.01.1970
      long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
      PrintFormat("Affare #%d DEAL_TIME_MSC=%i64 => %s",
                  last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
     }
   else
      PrintFormat("HistoryDealSelect() fallito per #%d. Codice Errore=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------------------+
//| Restituisce l'ultimo ticket nella cronistoria oppure -1                      |
//+------------------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- richiede la cronistoria per gli ultimi 7 giorni
   if(!GetTradeHistory(7))
     {
      //--- notifica sull'insuccesso della chiamata e restituisce -1
      Print(__FUNCTION__," HistorySelect() ha restituito false");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryOrdersTotal();
//--- lavora con gli ordini se ce ne sono
   if(deals>0)
     {
      Print("Affari = ",deals);
      first_deal=HistoryDealGetTicket(0);
      PrintFormat("first_deal = %d",first_deal);
      if(deals>1)
        {
         last_deal=HistoryDealGetTicket((int)deals-1);
         PrintFormat("last_deal = %d",last_deal);
         return last_deal;
        }
      return first_deal;
     }
//--- nessun affare trovato, restituito -1
   return -1;
  }
//+------------------------------------------------------------------------------+
//| Richiede la cronistoria per gli ultimi giorni e restituisce false in caso di falli
//+------------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- imposta il periodo di una settimana per richiedere la cronistoria di trading
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
```

```
   ResetLastError();
//---- fa una richiesta e controlla il risultato
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- cronitoria ricevuta con successo
   return true;
  }
```

Vedi anche

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Proprietà Affari

HistoryDealGetString

Restituisce la proprietà richiesta di un affare. La proprietà dell'affare deve essere di tipo stringa. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
string  HistoryDealGetString(
   ulong                      ticket_number,     // Ticket
   ENUM_DEAL_PROPERTY_STRING  property_id        // Proprietà Identificatore
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile di destinazione passata per riferimento dall ultimo parametro.

```
bool  HistoryDealGetString(
   ulong                      ticket_number,     // Ticket
   ENUM_DEAL_PROPERTY_STRING  property_id,       // Proprietà Identificatore
   string&                    string_var         // Qui si accetta il valore della pr
   );
```

Parametri

ticket_number

[in] Deal ticket.

property_id

[in] Identificatore della proprietà dell' affare. Il valore può essere uno dei valori dell'enumerazione ENUM_DEAL_PROPERTY_STRING.

string_var

[out] Variabile di tipo stringa che accetta il valore della proprietà richiesta.

Valore restituito

Valore di tipo stringa.

Nota

Attenzione a non confondere ordini, affari e posizioni. Ogni affare è il risultato dell'esecuzione di un ordine, ogni posizione è il risultato di riepilogo di uno o più affari.

Vedi anche

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Proprietà Affari

Segnali di Trade

Questo è il gruppo di funzioni consente di gestire segnali di trade. Le funzioni consentono:

- di ottenere informazioni su segnali di trade, disponibili per la copia,
- di ottenere e impostare i parametri di copia del segnale,
- di iscriversi e cancellarsi alla copia del segnale utilizzando le funzioni del linguaggio MQL5.

Funzione	Azione
SignalBaseGetDouble	Restituisce il valore della proprietà di tipo double per il segnale selezionato
SignalBaseGetInteger	Restituisce il valore della proprietà di tipo integer per il segnale selezionato
SignalBaseGetString	Restituisce il valore della proprietà di tipo string per il segnale selezionato
SignalBaseSelect	Seleziona un segnale dai segnali disponibili nel terminale per ulteriore lavorazione con esso
SignalBaseTotal	Restituisce la quantità totale di segnali, disponibile nel terminal
SignalInfoGetDouble	Restituisce il valore della proprietà di tipo double per i parametri di copia del segnale
SignalInfoGetInteger	Restituisce il valore della proprietà di tipo integer per i parametri di copia del segnale
SignalInfoGetString	Restituisce il valore della proprietà di tipo string per i parametri di copia del segnale
SignalInfoSetDouble	Imposta il valore della proprietà di tipo double per i parametri di copia del segnale
SignalInfoSetInteger	Imposta il valore della proprietà di tipo integer per i parametri di copia del segnale
SignalSubscribe	Sottoscrive il segnale di trading
SignalUnsubscribe	Annulla sottoscrizione

SignalBaseGetDouble

Restituisce il valore della proprietà di tipo double per il segnale selezionato.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // identificatore proprietà
   );
```

Parametri

property_id

[in] Identificatore proprietà del Segnale. Il valore può essere uno dei valori dell'enumerazione ENUM_SIGNAL_BASE_DOUBLE.

Valore restituito

Il valore della proprietà di tipo double del segnale selezionato.

SignalBaseGetInteger

Restituisce il valore della proprietà di tipo integer per il segnale selezionato.

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // identificatore proprietà
   );
```

Parametri

property_id

[in] Identificatore proprietà del Segnale. Il valore può essere uno dei valori dell' enumerazione ENUM_SIGNAL_BASE_INTEGER.

Valore restituito

Il valore della proprietà di tipo integer del segnale selezionato.

SignalBaseGetString

Restituisce il valore della proprietà di tipo string per il segnale selezionato.

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // identificatore proprietà
   );
```

Parametri

property_id

[in] Identificatore proprietà del Segnale. Il valore può essere uno dei valori dell' enumerazione ENUM_SIGNAL_BASE_STRING.

Valore restituito

Il valore della proprietà di tipo string del segnale selezionato.

SignalBaseSelect

Seleziona un segnale da segnali, disponibili nel terminale per ulteriore lavorazione con esso.

```
bool  SignalBaseSelect(
   int      index     // indice dei segnali
   );
```

Parametri

index

[in] L'indice del segnale in base ai segnali di trading.

Valore restituito

Restituisce vero se ha successo, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Esempio:

```
void OnStart()
  {
//--- ottiene il numero totale di segnali nel terminale
   int total=SignalBaseTotal();
//--- elabora tutti i segnali
   for(int i=0;i<total;i++)
     {
      //--- seleziona il segnale per indice
      if(SignalBaseSelect(i))
        {
         //--- ottiene le proprieta del segnale
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);           // id segnale
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);         // profitto in p
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // numero di sot
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);          // nome del segn
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);         // prezzo del se
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);      // signal currer
         //--- stampa tutti i segnali con profitto, gratuiti, con i sottoscritti
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,
        }
      else PrintFormat("Errore nella chiamata di SignalBaseSelect. Error code=%d",GetL
     }
  }
```

SignalBaseTotal

Restituisce la quantità totale di segnali, disponibile nel terminale.

```
int  SignalBaseTotal();
```

Valore restituito

La quantità totale di segnali, disponibile in terminale.

SignalInfoGetDouble

Restituisce il valore della proprietà di tipo double delle impostazioni di copia del segnale.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // identificatore proprietà
   );
```

Parametri

property_id

[in] Identificatore proprietà di parametri di copia del segnale. Il valore può essere uno dei valori dell'enumerazione ENUM_SIGNAL_INFO_DOUBLE.

Valore restituito

Il valore della proprietà di tipo double delle impostazioni di copia del segnale.

SignalInfoGetInteger

Restituisce il valore della proprietà di tipo integer delle impostazioni di copia del segnale.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identificatore proprietà
   );
```

Parametri

property_id

[in] Identificatore proprietà di parametri di copia del segnale. Il valore può essere uno dei valori dell'enumerazione ENUM_SIGNAL_INFO_INTEGER.

Valore restituito

Il valore della proprietà di tipo integer delle impostazioni di copia del segnale.

SignalInfoGetString

Restituisce il valore della proprietà di tipo string delle impostazioni di copia del segnale.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // identificatore proprietà
   );
```

Parametri

property_id

[in] Identificatore proprietà di parametri di copia del segnale. Il valore può essere uno dei valori dell'enumerazione ENUM_SIGNAL_INFO_STRING.

Valore restituito

Il valore della proprietà di tipo double delle impostazioni di copia del segnale.

SignalInfoSetDouble

Imposta il valore della proprietà di tipo double delle impostazioni di copia del segnale.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // identificatore proprietà
   double                       value            // nuovo valore
   );
```

Parametri

property_id

[in] Identificatore proprietà di parametri di copia del segnale. Il valore può essere uno dei valori dell'enumerazione ENUM_SIGNAL_INFO_DOUBLE.

valore

[in] Il valore delle proprietà di impostazioni di copia dei segnali.

Valore restituito

Restituisce true se la proprietà è stata modificata, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

SignalInfoSetInteger

Imposta il valore della proprietà di tipo integer delle impostazioni di copia del segnale.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identificatore proprietà
   long                         value            // nuovo valore
   );
```

Parametri

property_id

[in] Identificatore proprietà di parametri di copia del segnale. Il valore può essere uno dei valori dell'enumerazione ENUM_SIGNAL_INFO_INTEGER.

valore

[in] Il valore delle proprietà di impostazioni di copia dei segnali.

Valore restituito

Restituisce true se la proprietà è stata modificata, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

SignalSubscribe

Sottoscrizione al segnale di trading.

```
bool  SignalSubscribe(
   long     signal_id     // signal id
   );
```

Parametri

signal_id

[in] Identificatore Segnale.

Valore restituito

Restituisce true se la sottoscrizione ha avuto successo, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

SignalUnsubscribe

Annulla sottoscrizione

```
bool  SignalUnsubscribe();
```

Valore restituito

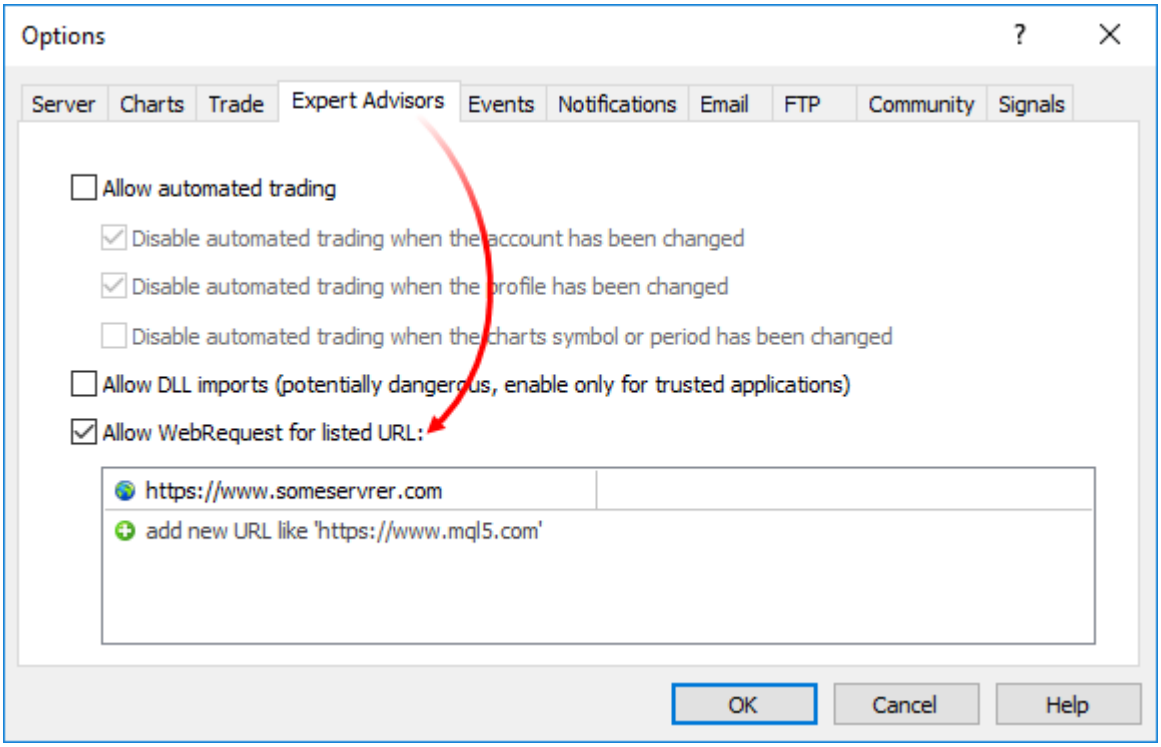
Restituisce true se sottoscrizione é stata annullata correttamente, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Funzioni di Rete

I programmi MQL5 possono scambiare dati con server remoti, nonché inviare notifiche push, e-mail e dati via FTP.

- Il gruppo di funzioni Socket * consente di stabilire una connessione TCP (incluso un TLS sicuro) con un host remoto tramite socket di sistema. Il principio di funzionamento è semplice: creare un socket, connettersi al server e iniziare lettura e scrittura dati.
- La funzione WebRequest è progettata per funzionare con risorse Web e consente di inviare facilmente richieste HTTP (inclusi GET e POST).
- SendFTP, SendMail e SendNotification sono funzioni più semplici per l'invio di file, e-mail e notifiche mobili.

Per la sicurezza dell'utente finale, l'elenco degli indirizzi IP consentiti è implementato dal lato client. L'elenco contiene gli indirizzi IP a cui il programma MQL5 può connettersi tramite le funzioni Socket* e WebRequest. Ad esempio, se il programma deve connettersi a https://www.someserver.com, questo indirizzo deve essere esplicitamente indicato da un utente terminale nell'elenco. Un indirizzo non può essere aggiunto a livello di codice.



Aggiunge un messaggio esplicito al programma MQL5 per notificare all'utente la necessità di una configurazione aggiuntiva. Puoi farlo via descrizione #property , Alert o Print.

Funzione	Azione
SocketCreate	Creare un socket con flag specificati e restituisce il relativo handle
SocketClose	Chiude un socket
SocketConnect	Si connette al server con il controllo del timeout
SocketIsConnected	Controlla se il socket è attualmente connesso

Funzione	Azione
SocketIsReadable	Ottiene un numero di byte che può essere letto da un socket
SocketIsWritable	Verifica se i dati possono essere scritti su un socket al momento attuale
SocketTimeouts	Imposta i timeout per la ricezione e l'invio di dati per un oggetto di sistema socket
SocketRead	Legge i dati da un socket
SocketSend	Scrive dati su un socket
SocketTlsHandshake	Avvia connessione TLS sicura (SSL) verso un host specificato tramite il protocollo TLS Handshake
SocketTlsCertificate	Ottiene dati sul certificato utilizzato per proteggere la connessione di rete
SocketTlsRead	Legge i dati dalla connessione TLS protetta
SocketTlsReadAvailable	Legge tutti i dati disponibili dalla connessione TLS protetta
SocketTlsSend	Invia dati tramite connessione TLS protetta
WebRequest	Invia una richiesta HTTP ad un server specificato
SendFTP	Invia un file ad un indirizzo specificato nella scheda FTP
SendMail	Invia un'email ad un indirizzo specificato nella scheda Email della finestra delle opzioni
SendNotification	Invia notifiche push ai terminali mobili i cui ID MetaQuotes sono specificati nella scheda Notifiche

SocketCreate

Crea un socket con flag specificati e restituisce il relativo handle.

```
int  SocketCreate(
   uint     flags         // flags
   );
```

Pparametri

flags

[in] Combinazione di flag che definiscono la modalità di funzionamento con un socket. Attualmente è supportato un solo flag: — SOCKET_DEFAULT.

Valore di Ritorno

In caso di successo di creazione di un socket, restituisce il relativo handle, altrimenti INVALID_HANDLE.

Note

Per liberare memoria del computer da un socket non utilizzato, chiamare SocketClose per questo.

È possibile creare un massimo di 128 socket da un programma MQL5. Se il limite viene superato, viene scritto l'errore 5271 (ERR_NETSOCKET_TOO_MANY_OPENED) _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
```

```
   if(len<0)
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Header risposta HTTP ricevuto:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
           }
         else
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
        }
      else
        {
         Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
        }
      //--- chiude un socket dopo averlo usato
      SocketClose(socket);
     }
   else
      Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketClose

Chiude un socket.

```
bool  SocketClose(
   const int  socket      // handle del socket
   );
```

Pparametri

socket

[in] Handle del socket da chiudere. L'handle viene restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

Valore di Ritorno

Restituisce true se ha successo, altrimenti false.

Nota

Se una connessione tramite SocketConnect è stata precedentemente creata per un socket, viene interrotta.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
//|                                                  SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
```

```
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Header risposta HTTP ricevuto:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
//+------------------------------------------------------------------+
void OnStart()
```

```
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
           }
         else
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
        }
      else
        {
         Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
        }
      //--- chiude un socket dopo averlo usato
      SocketClose(socket);
     }
   else
      Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketConnect

Connette al server con il controllo del timeout.

```
bool  SocketConnect(
   int           socket,              // socket
   const string  server,              // indirizzo di connessione
   uint          port,                // porta di connessione
   uint          timeout_receive_ms   // timeout connesione
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

server

[in] Nome del dominio del server a cui ci si vuole connettere, o il suo indirizzo IP.

port

[in] Numero della porta di connessione.

timeout_receive_ms

[in] Timeout della connessione in millisecondi. Se la connessione non viene stabilita entro tale intervallo di tempo, i tentativi vengono interrotti.

Valore di Ritorno

Se la connessione ha esito positivo, restituisce true, altrimenti false.

Nota

L'indirizzo di connessione deve essere aggiunto all'elenco di quelli consentiti sul lato client (Strumenti\Opzioni\Expert Advisor).

Se la connessione non riesce, viene scritto l'errore 5272 (ERR_NETSOCKET_CANNOT_CONNECT) in _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
```

```
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Header risposta HTTP ricevuto:");
```

```
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
           }
         else
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
        }
      else
        {
```

```
         Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
        }
      //--- chiude un socket dopo averlo usato
      SocketClose(socket);
     }
   else
      Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketIsConnected

Controlla se il socket è attualmente connesso.

```
bool  SocketIsConnected(
   const int  socket      // handle del socket
   );
```

Pparametri

socket

[in] Handle del socket restituito dalla funzione SocketCreate(). Quando viene passato un handle errato _LastError, si attiva l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valore di Ritorno

Restituisce true se il socket è connesso, altrimenti - false.

Nota

La funzione SocketIsConnected() consente di verificare lo stato corrente della connessione socket.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Guarda anche

SocketConnect, SocketIsWritable, SocketCreate, SocketClose

SocketIsReadable

Ottieni un numero di byte che può essere letto da un socket.

```
uint  SocketIsReadable(
   const int  socket      // handle del socket
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato _LastError, si attiva l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valore di Ritorno

Numero di byte che possono essere letti. In caso di errore, viene restituito 0.

Nota

Se si verifica un errore su un socket di sistema durante l'esecuzione della funzione, la connessione stabilita tramite SocketConnect viene interrotta.

Prima di chiamare SocketRead, controlla se il socket contiene dati per la lettura. Altrimenti, se non ci sono dati, la funzione SocketRead attende i dati entro il timeout_ms ritardando l'esecuzione del programma.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
```

```
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Header risposta HTTP ricevuto:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
           }
         else
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
        }
      else
        {
         Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
        }
      //--- chiude un socket dopo averlo usato
      SocketClose(socket);
     }
   else
      Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketIsWritable

Verifica se i dati possono essere scritti su un socket al momento attuale.

```
bool  SocketIsWritable(
   const int  socket      // handle del socket
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

Valore di Ritorno

Restituisce true se la scrittura è possibile, altrimenti false.

Nota

Questa funzione ti permette di verificare se è possibile scrivere dati su un socket in questo momento.

Se si verifica un errore su un socket di sistema durante l'esecuzione della funzione, la connessione stabilita tramite SocketConnect viene interrotta.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

SocketTimeouts

Imposta i timeout per la ricezione e l'invio di dati per un oggetto di sistema socket.

```
bool  SocketTimeouts(
   int           socket,               // socket
   uint          timeout_send_ms,      // timeout di invio dati
   uint          timeout_receive_ms    // timeout ottenimento dati
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

timeout_send_ms

[in] Timeout di invio dati, in millisecondi.

timeout_receive_ms

[in] Timeout di ottenimento dati, in millisecondi.

Valore di Ritorno

Restituisce true se ha successo, altrimenti false.

Nota

Non confondere i timeout degli oggetti di sistema con quelli impostati durante la lettura dei dati tramite SocketRead. SocketTimeout imposta una volta i timeout per un oggetto socket nel sistema operativo. Questi timeout devono essere applicati a tutte le funzioni per la lettura e l'invio di dati tramite questo socket. In SocketRead, il timeout è impostato per una determinata operazione di lettura dei dati.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

SocketRead

Legge i dati da un socket.

```
int  SocketRead(
   int           socket,              // socket
   uchar&        buffer[],            // buffer per la lettura dei dati dal socket
   uint          buffer_maxlen,       // numero di byte da leggere
uinttimeout_ms// timeout lettura
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato _LastError, si attiva l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Riferimento all'array di tipo uchar in cui vengono letti i dati. La dimensione dinamica dell'array è aumentata dal numero di byte letti. La dimensione dell'array non può superare INT_MAX (2147483647).

buffer_maxlen

[in] Numero di byte da leggere nell'array*buffer[]*. I dati non adattati all'array rimangono nel socket. Possono essere ricevuti dalla prossima chiamata SocketRead. *buffer_maxlen* cannot exceed INT_MAX (2147483647).

timeout_ms

[in] Tempo di lettura dei dati in millisecondi. Se i dati non vengono ottenuti entro questo tempo, i tentativi vengono interrotti e la funzione restituisce -1.

Valore di Ritorno

Se ha successo, restituisce il numero di byte letti. In caso di errore, viene restituito -1.

Nota

Se si verifica un errore su un socket di sistema durante l'esecuzione della funzione, la connessione stabilita tramite SocketConnect viene interrotta.

In caso di errore di lettura dei dati, l'errore 5273 (ERR_NETSOCKET_IO_ERROR) viene scritto in _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

```
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
```

```
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Header risposta HTTP ricevuto:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            }
          }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
           }
         else
```

```
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
         }
       else
         {
          Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
         }
       //--- chiude un socket dopo averlo usato
       SocketClose(socket);
      }
    else
       Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Guarda anche

SocketTimeouts, MathSwap

SocketSend

Scrivi dati su un socket.

```
int  SocketSend(
   int           socket,              // socket
   const uchar&  buffer[],            // buffer dati
   uint          buffer_len           // grandezza buffer
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

buffer

[in] Riferimento array di tipo uchar con i dati da inviare al socket.

buffer_len

[in] grandezza dell'array 'buffer'.

Valore di Ritorno

In caso di successo, restituisce il numero di byte scritti su un socket. In caso di errore, viene restituito -1.

Nota

Se si verifica un errore su un socket di sistema durante l'esecuzione della funzione, la connessione stabilita tramite SocketConnect viene interrotta.

In caso di errore di scrittura dei dati, viene scritto l'errore 5273 (ERR_NETSOCKET_IO_ERROR) in _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
#property script_show_inputs

input string Address="www.mql5.com";
```

```
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Header risposta HTTP ricevuto:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
```

```
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
           }
         else
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
        }
      else
        {
         Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
        }
      //--- chiude un socket dopo averlo usato
```

```
      SocketClose(socket);
     }
   else
      Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Guarda anche

SocketTimeouts, MathSwap, StringToCharArray

SocketTlsHandshake

Avvia connessione TLS sicura (SSL) ad un host specificato tramite il protocollo TLS Handshake. Durante l'handshake, un client ed un server concordano sui parametri di connessione: versione del protocollo applicata e metodo di crittografia dei dati.

```
bool  SocketTlsHandshake(
   int           socket,               // socket
   const string  host                  // indirizzo dell'host
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

host

[in] L'indirizzo di un host con cui è stabilita una connessione sicura.

Valore di Ritorno

Restituisce true se ha successo, altrimenti false.

Note

Prima di una connessione sicura, il programma dovrebbe stabilire una connessione TCP standard con l'host che utilizza SocketConnect.

Se la connessione protetta non riesce, viene scritto l'errore 5274 (ERR_NETSOCKET_HANDSHAKE_FAILED) in _LastError.

Non è necessario chiamare la funzione quando vi è collegamento alla porta 443. Questa è una porta TCP standard utilizzata per connessioni TLS (SSL) sicure.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

SocketTlsCertificate

Ottieni dati sul certificato utilizzato per proteggere la connessione di rete.

```
int  SocketTlsCertificate(
   int           socket,               // socket
   string&       subject,              // proprietario del certificato
   string&       issuer,               // emittente del certificato
   string&       serial,               // numero di serie del certificato
stringa& identificazione personale,// impronta(print) del certificato
   datetime&     expiration            // scadenza del certificato
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

subject

[in] Nome del proprietario del certificato. Corrisponde al campo Subject.

issuer

[in] Nome dell'emittente del certificato. Corrisponde al campo Issuer.

serial

[in] Numero di serie del certificato. Corrisponde al campo SerialNumber.

thumbprint

[in] Print(impronta) del Certificato. Corrisponde all'hash SHA-1 dell'intero file del certificato (tutti i campi inclusa la firma dell'emittente).

expiration

[in] Data di scadenza del certificato in formato datetime.

Valore di Ritorno

Restituisce true se ha successo, altrimenti false.

Nota

I dati del certificato possono essere richiesti solo dopo aver stabilito una connessione sicura utilizzando SocketTlsHandshake.

In caso di errore di ottenimento di un certificato, viene scritto l'errore 5275 (ERR_NETSOCKET_NO_CERTIFICATE) in _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
```

```
//|                                                  SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
```

```
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Header risposta HTTP ricevuto:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
```

```
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
            }
         else
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
        }
      else
        {
         Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
        }
      //--- chiude un socket dopo averlo usato
      SocketClose(socket);
     }
   else
      Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketTlsRead

Legge i dati dalla connessione TLS protetta.

```
int  SocketTlsRead(
   int           socket,              // socket
   uchar&        buffer[],            // buffer per la lettura dei dati dal socket
   uint          buffer_maxlen        // numero di byte da leggere
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato _LastError, si attiva l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Riferimento all'array di tipo uchar in cui vengono letti i dati. La dimensione dinamica dell'array è aumentata dal numero di byte letti. La dimensione dell'array non può superare INT_MAX (2147483647).

buffer_maxlen

[in] Numero di byte da leggere nell'array*buffer[]*. I dati non adattati all'array rimangono nel socket. Possono essere ricevuti dalla prossima chiamata SocketTLSRead. *buffer_maxlen* cannot exceed INT_MAX (2147483647).

Valore di Ritorno

Se ha successo, restituisce il numero di byte letti. In caso di errore, viene restituito -1.

Nota

Se si verifica un errore su un socket di sistema durante l'esecuzione della funzione, la connessione stabilita tramite SocketConnect viene interrotta.

La funzione viene eseguita finché non riceve la quantità specificata di dati o viene raggiunto il timeout (SocketTimeouts).

In caso di errore di lettura dei dati, l'errore 5273 (ERR_NETSOCKET_IO_ERROR) viene scritto in _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Esempio:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
```

```
#property version     "1.00"
#property description "Aggiungi indirizzo all'elenco di quelli consentiti nelle impost
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Invia comando al server                                          |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se viene utilizzata una connessione TLS sicura tramite la porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se viene utilizzata la connessione TCP standard
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Legge la risposta del server                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- legge i dati dai socket finché sono ancora presenti ma non più lunghi del timeou
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         // --- vari comandi di lettura a seconda che la connessione sia sicura o meno
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         // --- analizza la risposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            // --- stampa solo l'intestazione della risposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
```

```
               {
                Print("Header risposta HTTP ricevuto:");
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Funzione Start del programma di script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- controlla l'handle
   if(socket!=INVALID_HANDLE)
     {
      //--- connetti se tutto va bene
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Connessione stabilita a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         // --- se la connessione è protetta dal certificato, visualizza i suoi dati
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:  ",subject);
            Print("   Issuer:  ",issuer);
            Print("   Number:     ",serial);
            Print("   Print: ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         // --- invia richiesta GET al server
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request inviata");
            // --- leggi la risposta
            if(!HTTPRecv(socket,1000))
               Print("Impossibile ottenere una risposta, errore ",GetLastError());
           }
         else
            Print("Impossibile inviare la richiesta GET, errore",GetLastError());
        }
```

```
      else
        {
         Print("Connessione a ",Address,":",Port," fallita, errore ",GetLastError());
        }
      //--- chiude un socket dopo averlo usato
      SocketClose(socket);
     }
   else
      Print("Impossibile creare un socket, errore ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Guarda anche

SocketTimeouts, MathSwap

SocketTlsReadAvailable

Leggi tutti i dati disponibili dalla connessione TLS protetta.

```
int  SocketTlsReadAvailable(
   int           socket,              // socket
   uchar&        buffer[],            // buffer per la lettura dei dati dal socket
   const uint    buffer_maxlen        // numero di byte da leggere
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato _LastError, si attiva l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Riferimento all'array di tipo uchar in cui vengono letti i dati. La dimensione dinamica dell'array è aumentata dal numero di byte letti. La dimensione dell'array non può superare INT_MAX (2147483647).

buffer_maxlen

[in] Numero di byte da leggere sull'array del buffer[]. I dati non adattati all'array rimangono nel socket. Possono essere ricevuti dalla prossima chiamata SocketTlsReadAvailable oSocketTlsRead. buffer_maxlen cannot exceed INT_MAX (2147483647).

Valore di Ritorno

Se ha successo, restituisce il numero di byte letti. In caso di errore, viene restituito -1.

Nota

Se si verifica un errore su un socket di sistema durante l'esecuzione della funzione, la connessione stabilita tramite SocketConnect viene interrotta.

In caso di errore di lettura dei dati, l'errore 5273 (ERR_NETSOCKET_IO_ERROR) viene scritto in _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Guarda anche

SocketTimeouts, MathSwap

SocketTlsSend

Invia dati tramite connessione TLS protetta.

```
int  SocketTlsSend(
   int           socket,              // socket
   const uchar&  buffer[],            // buffer dati
   uint          buffer_len           // grandezza buffer
   );
```

Pparametri

socket

[in] Il socket handle restituito dalla funzione SocketCreate. Quando viene passato un handle errato, viene scritto l'errore 5270 (ERR_NETSOCKET_INVALIDHANDLE) _LastError.

buffer

[in] Riferimento all'array di tipo uchar con i dati da inviare.

buffer_len

[in] grandezza dell'array 'buffer'.

Valore di Ritorno

In caso di successo, restituisce il numero di byte scritti su un socket. In caso di errore, viene restituito -1.

Nota

Se si verifica un errore su un socket di sistema durante l'esecuzione della funzione, la connessione stabilita tramite SocketConnect viene interrotta.

In caso di errore di scrittura dei dati, viene scritto l'errore 5273 (ERR_NETSOCKET_IO_ERROR) in _LastError.

La funzione può essere chiamata solo da Expert Advisors e scripts, poiché vengono eseguiti nei relativi thread di esecuzione. Se si chiama da un indicatore, GetLastError() restituisce l'errore 4014 - "Function is not allowed for call (la funzione non è consentita per la chiamata)".

Guarda anche

SocketTimeouts, MathSwap, StringToCharArray

WebRequest

La funzione invia una richiesta HTTP ad un server specificato. La funzione ha due versioni:

1. **Invio richieste semplici** di tipo "key=value" usando l'header Content-Type: application/x-www-form-urlencoded.

```
int  WebRequest(
   const string      method,           // metodo HTTP
   const string      url,              // URL
   const string      cookie,           // cookie
   const string      referer,          // referer
   int               timeout,          // timeout
   const char        &data[],          // l'array del corpo del messaggio HTTP
   int               data_size,        // grandezza array data[] in bytes
   char              &result[],        // un array contenente i dati di risposta del s
   string            &result_headers   // headers della risposta del server
   );
```

2. **Invio di una richiesta di qualunque tipo** specificando il personalizzato set di headers per una interazione più flessibile con vari servizi Web.

```
int  WebRequest(
   const string      method,           // metodo HTTP
   const string      url,              // URL
   const string      headers,          // headers
   int               timeout,          // timeout
   const char        &data[],          // l'array del corpo del messaggio HTTP
   char              &result[],        // un array contenente i dati di risposta del s
   string            &result_headers   // headers della risposta del server
   );
```

Parametri

metodo

[in] metodo HTTP.

url

[in] URL.

headers

[in] Richiedi intestazioni di tipo "chiave: valore", separate da un'interruzione di riga "\\ r \\ n".

cookie

[in] Valore del cookie.

referente

[in] Valore dell'intestazione Referer della richiesta HTTP.

timeout

[in] Timeout in millisecondi.

data[]

[in] Array di dati del corpo del messaggio HTTP.

data_size

[in] Dimensione dell'array di dati[].

result[]

[out] Un array contenente dati di risposta del server.

result_headers

[out] Intestazioni di risposta del server.

Valore restituito

Codice di risposta del server HTTP o -1 per un errore.

Nota

Per utilizzare la funzione WebRequest(), aggiungere gli indirizzi dei server richiesti nell'elenco di URL consentiti nella scheda "Expert Advisors" della finestra "Opzioni". La porta del server viene selezionata automaticamente sulla base del protocollo specificato - 80 per "http://" e 443 per "https://".

La funzione WebRequest() è sincrona, il che significa che interrompe l'esecuzione del programma e attende la risposta dal server richiesto. Poiché i ritardi nella ricezione di una risposta possono essere elevati, la funzione non è disponibile per le chiamate dagli indicatori, poiché gli indicatori vengono eseguiti in un thread comune condiviso da tutti gli indicatori e charts su un unico simbolo. Il ritardo delle prestazioni dell'indicatore su uno dei charts di un simbolo può interrompere l'aggiornamento di tutti i charts dello stesso simbolo.

La funzione può essere chiamata solo da Expert Advisors e script, poiché vengono eseguiti nei relativi thread di esecuzione. Se provi a chiamare la funzione da un indicatore, GetLastError() restituirà l'errore 4014 - "Funzione non consentita".

WebRequest() non può essere eseguito nel Tester di strategia.

Esempio:

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- Per abilitare l'accesso al server, è necessario aggiungere l'URL "https://finan
//--- alla lista di URL consentiti (Main Menu->Tools->Options, scheda "Expert Advisors
// --- Reimpostazione dell'ultimo codice di errore
   ResetLastError();
// --- Download di una pagina html da Yahoo Finance
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Errore in WebRequest. Error code  =",GetLastError());
      //--- Forse l'URL non è elencato, mostra un messaggio sulla necessità di aggiun
```

```
      MessageBox("Aggiungi l'indirizzo ' "+url+" ' alla lista di URL consentiti sulla
     }
   else
     {
      if(res==200)
        {
         //--- Download riuscito
         PrintFormat("Il file è stato scaricato con successo, Dimensione File %d byte
         //PrintFormat("Server headers: %s",headers);
         //--- Salvataggio dati in un file
         int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
         if(filehandle!=INVALID_HANDLE)
           {
            //--- Salvataggio contenuto dell'array result[] in un file
            FileWriteArray(filehandle,result,0,ArraySize(result));
            //--- Chiusura file
            FileClose(filehandle);
           }
         else
            Print("Errore in FileOpen. Error code =",GetLastError());
        }
      else
         PrintFormat("Downloading '%s' fallito, codice errore %d",url,res);
     }
  }
```

SendFTP

Invia un file all'indirizzo, specificato nella finestra di impostazione della scheda "FTP".

```
bool  SendFTP(
   string  filename,          // file da spedire con ftp
   string  ftp_path=NULL      // ftp catalog
   );
```

Parametri

filename

[in] Nome del file spedito.

ftp_path=NULL

[in] catalogo FTP. Se una directory non è specificata, viene utilizzata la directory descritta nelle impostazioni.

Valore restituito

In caso di fallimento restituisce 'false'.

Nota

Il file inviato deve essere posizionato nella cartella *terminal_directory\MQL5\files* o sue relative sottocartelle. L'invio non viene eseguito se l'indirizzo FTP e/o la password di accesso non sono specificate nelle impostazioni.

SendFTP() function does not work in the Strategy Tester.

SendMail

Invia una e-mail all'indirizzo specificato nella finestra delle impostazioni della scheda "Email".

```
bool  SendMail(
   string  subject,        // header
   string  some_text       // testo email
   );
```

Parametri

soggetto

[in] header Email.

un_qualche_testo

[in] Corpo dell'e-mail.

Valore restituito

true - se una e-mail viene messa in coda di invio, in caso contrario - false.

Nota

L'invio può essere vietato nelle impostazioni, l' indirizzo e-mail può essere omesso, pure. Per le informazioni di errore chiamare GetLastError().

SendMail() function does not work in the Strategy Tester.

SendNotification

Invia le notifiche push per i terminali mobili, i cui MetaQuotes IDs sono specificati nella scheda "Notifiche".

```
bool  SendNotification(
   string  text          // Testo della notifica
   );
```

Parametri

text

[in] Il testo della notifica. La lunghezza del messaggio non deve superare i 255 caratteri.

Valore restituito

true se la notifica è stata inviata con successo dal terminale, in caso di fallimento restituisce false. Durante il controllo dopo una notifica push fallita, GetLastError() può restituire uno dei seguenti errori:

- 4515 - ERR_NOTIFICATION_SEND_FAILED,
- 4516 - ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 - ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 - ERR_NOTIFICATION_TOO_FREQUENT.

Nota

Severe restrizioni d'uso sono impostate per la funzione SendNotification(): non più di 2 chiamate al secondo e non più di 10 chiamate al minuto. Il monitoraggio della frequenza di utilizzo è dinamico. La funzione può essere disattivata in caso di violazione restrizione.

SendNotification() function does not work in the Strategy Tester.

Global Variables of the Client Terminal

Vi è una serie di funzioni di gruppo per l'utilizzo di variabili globali.

Le variabili globali del terminale client non devono essere confuse con le variabili dichiarate nell'ambito globale del programma mql5.

Le variabili globali sono conservate nel terminale client per 4 settimane dopo l'ultimo accesso, poi verranno cancellate automaticamente. Un accesso ad una variabile è non solo l'impostazione di un nuovo valore, ma la lettura del valore della variabile globale, anche.

Le variabili globali del terminale client sono accessibili simultaneamente da tutti i programmi mql5 avviati nel terminale client.

Funzione	Azione
GlobalVariableCheck	Controlla l'esistenza di una variabile globale con il nome specificato
GlobalVariableTime	Restituisce l'orario dell'ultimo accesso alla variabile globale
GlobalVariableDel	Elimina una variabile globale
GlobalVariableGet	Restituisce il valore di una variabile globale
GlobalVariableName	Restituisce il nome di una variabile globale dal suo numero ordinale nell'elenco delle variabili globali
GlobalVariableSet	Imposta il nuovo valore in una variabile globale
GlobalVariablesFlush	Salva forzatamente contenuto di tutte le variabili globali nel disco
GlobalVariableTemp	Imposta il nuovo valore in una variabile globale, che esiste solo nella sessione corrente del terminale
GlobalVariableSetOnCondition	Imposta il nuovo valore dell'esistente variabile globale per condizione
GlobalVariablesDeleteAll	Elimina le variabili globali con il prefisso specificato nel loro nome
GlobalVariablesTotal	Restituisce il numero totale di variabili globali

GlobalVariableCheck

Controlla l'esistenza di una variabile globale con il nome specificato

```
bool  GlobalVariableCheck(
   string  name      // Nome della variabile globale
   );
```

Parametri

name

[in] Nome della variabile globale.

Valore restituito

Restituisce true se la variabile globale esiste, altrimenti restituisce false.

Le variabili globali presenti nel terminale client per 4 settimane dopo il loro ultimo uso, vengono dunque cancellati automaticamente.

Vedi anche

GlobalVariableTime()

GlobalVariableTime

Restituisce l'orario dell' ultimo accesso alla variabile globale.

```
datetime  GlobalVariableTime(
   string  name      // nome
   );
```

Parametri

name

[in] Nome della variabile globale.

Valore restituito

La funzione restituisce l'ora dell'ultimo accesso alla variabile globale specificata. La chiamata di una variabile per ottenere il valore è anche considerata come un accesso ad essa. Per ottenere dettaggli sull'errore, chiamare la funzione GetLastError().

Nota

Le variabili globali esistono nel terminale client per 4 settimane da quando sono state chiamate l'ultima volta. Dopo di che vengono automaticamente cancellati.

Vedi anche

GlobalVariableCheck()

GlobalVariableDel

Elimina una variabile globale del terminale client.

```
bool  GlobalVariableDel(
   string  name      // Nome della variabile globale
   );
```

Parametri

name

[in] Nome della variabile globale.

Valore restituito

In caso di successo, la funzione restituisce true, altrimenti restituisce false. Per ottenere informazioni sull' errore è necessario chiamare la funzione GetLastError().

Nota

Le variabili globali presenti nel terminale client per 4 settimane dopo il loro ultimo uso, vengono dunque cancellati automaticamente.

GlobalVariableGet

Restituisce il valore di una variabile globale esistente del terminale client. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della variabile globale.

```
double  GlobalVariableGet(
   string  name      // Nome della variabile globale
   );
```

2. Restituisce true o false a seconda del successo dell'esecuzione della funzione. In caso di successo, la variabile globale del terminale client è posta in una variabile passata per riferimento nel secondo parametro.

```
bool  GlobalVariableGet(
   string  name,              // Nome variabile globale
   double& double_var         // Questa variabile conterrà il valore della variabile g
   );
```

Parametri

name

[in] Nome della variabile globale.

double_var

[out] Variabile target di tipo double, che accetta il valore memorizzato in una variabile globale del terminale client.

Valore restituito

Il valore della variabile globale esistente oppure 0 in caso di errore. Per ulteriori informazioni sull'errore, chiamare GetLastError().

Nota

Le variabili globali presenti nel terminale client per 4 settimane dopo il loro ultimo uso, vengono dunque cancellati automaticamente.

GlobalVariableName

Restituisce il nome di una variabile globale dal suo numero ordinale.

```
string  GlobalVariableName(
   int  index      // Numero della variabile globale nell'elenco delle variabili globa
   );
```

Parametri

index

[n] Numero di sequenza nella lista delle variabili globali. Essa deve essere maggiore o uguale a 0 e minore di GlobalVariablesTotal().

Valore restituito

Il nome della variabile globale per il suo numero ordinale nell'elenco delle variabili globali. Per più dettagli riguardo lerrore, call GetLastError().

Nota

Le variabili globali presenti nel terminale client per 4 settimane dopo il loro ultimo uso, vengono dunque cancellati automaticamente.

GlobalVariableSet

Imposta un nuovo valore per una variabile globale. Se la variabile non esiste, il sistema crea una nuova variabile globale.

```
datetime  GlobalVariableSet(
   string  name,       // Nome della variabile globale
   double  value       // Valore da impostare
   );
```

Parametri

name

[in] Nome della variabile globale.

valore

[in] Il nuovo valore numerico.

Valore restituito

In caso di successo, la funzione restituisce l'ora dell'ultima modifica, altrimenti 0. Per maggiori dettagli sull'errore, chiamare GetLastError().

Nota

A global variable name should not exceed 63 characters. Le variabili globali presenti nel terminale client per 4 settimane dopo il loro ultimo uso, vengono dunque cancellati automaticamente.

GlobalVariablesFlush

Salva forzatamente contenuto di tutte le variabili globali nel disco.

```
void  GlobalVariablesFlush();
```

Valore restituito

Nessun valore restituito.

Nota

Il terminale scrive tutte le variabili globali quando il lavoro è finito, ma i dati possono andare persi in caso di guasto improvviso del funzionamento del computer. Questa funzione permette di controllare in modo indipendente il processo di salvataggio delle variabili globali in caso di emergenza.

GlobalVariableTemp

La funzione tenta di creare una variabile globale temporanea. Se la variabile non esiste, il sistema crea una nuova variabile globale temporanea.

```
bool  GlobalVariableTemp(
   string  name      // Nome della variabile globale
   );
```

Parametri

name

[in] Il nome della variabile temporanea globale.

Valore restituito

In caso di successo, la funzione restituisce true, in caso contrario - false. Per dettagli sull' errore, è necessario chiamare la funzione GetLastError().

Nota

Le variabili globali temporanee sono uniche mentre il terminale client è in esecuzione, dopo la chiusura del terminale vengono automaticamente eliminate. Notare che durante l'esecuzione di GlobalVariablesFlush() le variabili globali temporanee non vengono scritte nel disco.

Dopo che una variabile globale temporanea è stato creata, è possibile accedere e modificare la stessa come variabile globale del terminale client.

GlobalVariableSetOnCondition

Imposta il nuovo valore della variabile globale esistente se il valore corrente è uguale al terzo parametro check_value. Se non vi è nessuna variabile globale, la funzione genererà un errore ERR_GLOBALVARIABLE_NOT_FOUND (4501) e restituisce false.

```
bool  GlobalVariableSetOnCondition(
   string  name,            // Nome della variabile globale
   double  value,           // Nuovo valore per la variabile se la condizione è true
   double  check_value      // Condizione di controllo valore
   );
```

Parametri

name

[in] Il nome di una variabile globale.

valore

[in] New value.

check_value

[in] Il valore per controllare il valore corrente della variabile globale.

Valore restituito

In caso di successo, la funzione restituisce true, altrimenti restituisce false. Per i dettagli sull'errore chiamare GetLastError(). Se il valore corrente della variabile globale è diverso da check_value, la funzione restituisce false.

Nota

La funzione fornisce l'accesso atomico per la variabile globale, in modo che possa essere utilizzata per fornire un mutex all' interazione dei diversi Expert Advisors che lavorano simultaneamente all'interno del terminale client.

GlobalVariablesDeleteAll

Elimina le variabili globali del terminale cliente.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL,     // Tutte le variabili globali con nomi che iniziar
   datetime   limit_data=0          // Tutte le variabili globali che sono state modi
   );
```

Parametri

prefix_name=NULL

[in] Nome prefisso delle variabili globali da rimuovere. Se si specifica un prefisso NULL o una stringa vuota, allora tutte le variabili che soddisfano il criterio dati, verranno eliminate.

limit_data=0

[in] Data per selezionare le variabili globali al momento della loro ultima modifica. La funzione rimuove le variabili globali, che sono state cambiati prima di questa data. Se il parametro è zero, allora tutte le variabili che soddisfano il primo criterio (prefisso) vengono soppresse.

Valore restituito

Il numero delle variabili eliminate.

Nota

Se entrambe le opzioni sono uguali a zero (prefix_name = NULL e limit_data = 0), allora la funzione elimina tutte le variabili globali del terminale. Se entrambi i parametri vengono specificati, allora elimina le variabili globali corrispondenti ad entrambi i parametri.

Le variabili globali presenti nel terminale client per 4 settimane dopo il loro ultimo uso, poi vengono cancellate automaticamente.

GlobalVariablesTotal

Restituisce il numero totale di variabili globali del terminale cliente.

```
int  GlobalVariablesTotal();
```

Valore restituito

Numero di variabili globali.

Nota

Le variabili globali presenti nel terminale cliente per 4 settimane dopo il loro ultimo uso, poi vengono cancellate automaticamente. La chiamata di una variabile non avviene solo impostando un nuovo valore, ma anche con la lettura del valore della variabile globale.

Funzioni con i File

Questo è un gruppo di funzioni per lavorare con file.

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Ci sono due directory (con le sottodirectory) in cui i file di lavoro possono essere ubicati:

- terminal_data_folder\MQL5\FILES\ (nel menu del terminale scegliere di visualizzare "File" - "Apri la directory dei dati");
- la cartella comune per tutti i terminali installati in un computer - di solito si trova nella directory C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files.

C'è un metodo di programma per ottenere nomi di questi cataloghi utilizzando la funzione TerminalInfoString(), utilizzando l'enumerazione ENUM_TERMINAL_INFO_STRING

```
//--- Cartella che memorizza i dati del terminale
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- Cartella comune per tutti i terminali client
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Lavorare con i file da altre directory è proibito.

Le funzioni dei file permettono di lavorare con i cosiddetti "named pipe". Per fare questo, è sufficiente chiamare la funzione FileOpen() con parametri appropriati.

Funzione	Azione
FileSelectDialog	Crea una finestra di dialogo di apertura/creazione di file o cartelle
FileFindFirst	Avvia la ricerca di file in una directory secondo il filtro specificato
FileFindNext	Continua la ricerca iniziata dalla funzione FileFindFirst()
FileFindClose	Chiude l'handle di ricerca
FileOpen	Apre un file con nome e flag specificati
FileDelete	Elimina un file specificato
FileFlush	Scrive su un disco tutti i dati rimasti nel buffer input/output del file
FileGetInteger	Ottiene una proprietà integer di un file
FileIsEnding	Definisce la fine di un file nel processo di lettura
FileIsLineEndin	Definisce la fine di una riga in un file di testo nel processo di lettura
FileClose	Chiude un file aperto in precedenza
FileIsExist	Controlla l'esistenza di un file
FileCopy	Copia il file originale da una cartella locale o condivisa in un altro file
FileMove	Sposta o rinomina un file

Funzione	Azione
FileReadArray	Legge array di qualsiasi tipo ad eccezione di string da file di tipo BIN
FileReadBool	Legge dal file di tipo CSV una stringa a partire dalla posizione corrente fino ad un delimitatore (o fino alla fine di una riga di testo), e converte la stringa di lettura per un valore di tipo bool
FileReadDatetime	Legge dal file di tipo CSV una stringa in uno dei formati: "AAAA.MM.GG. HH:MM:SS", "AAAA.MM.GG." o "HH:MM:SS" - e la converte in un valore datetime
FileReadDouble	Legge un valore double dalla posizione corrente del puntatore del file
FileReadFloat	Legge un valore float dalla posizione corrente del puntatore del file
FileReadInteger	Legge valori int, short o char, dalla posizione corrente del puntatore del file
FileReadLong	Legge un valore di tipo long dalla posizione corrente del puntatore del file
FileReadNumber	Legge dal file di tipo CSV una stringa a partire dalla posizione corrente fino ad un delimitatore (o fino alla fine di una riga di testo), e converte la stringa di lettura in valore double
FileReadString	Legge una stringa dalla posizione corrente di un puntatore ad un file da un file
FileReadStruct	Legge il contenuto di un file binario in una struttura passato come parametro, dalla posizione corrente del puntatore del file
FileSeek	Sposta la posizione del puntatore del file di un determinato numero di byte rispetto alla posizione specificata
FileSize	Restituisce la grandezza del corrispondente file aperto
FileTell	Restituisce la posizione corrente del puntatore del file del corrispondente file aperto
FileWrite	Scrive dati in un file di tipo CSV o TXT
FileWriteArray	Scrive array di qualsiasi tipo ad eccezione di string in un file di tipo BIN
FileWriteDouble	Scrive il valore di tipo double dalla posizione corrente di un puntatore di un file in un file binario
FileWriteFloat	Scrive il valore di tipo float dalla posizione corrente di un puntatore di un file in un file binario
FileWriteInteger	Scrive il valore di tipo int dalla posizione corrente di un puntatore di un file in un file binario
FileWriteLong	Scrive il valore di tipo long dalla posizione corrente di un puntatore di un file in un file binario

Funzione	Azione
FileWriteString	Scrive il valore di un parametro string in un file BIN o TXT a partire dalla posizione corrente del puntatore del file
FileWriteStruct	Scrive il contenuto di una struttura passata come parametro in un file binario, a partire dalla posizione corrente del puntatore del file
FileLoad	Legge tutti i dati di un file binario specificato in un array passato, di tipo numerico o strutture semplici
FileSave	Scrive in un file binario tutti gli elementi di un array passato come parametro
FolderCreate	Crea una cartella nella directory dei file
FolderDelete	Rimuove una directory selezionata. Se la cartella non è vuota, allora non può essere rimossa
FolderClean	Elimina tutti i file presenti nella cartella specificata

Se il file viene aperto per la scrittura usando FileOpen(), tutte le sottocartelle specificate nel percorso verranno create se queste non ci sono .

FileSelectDialog

Crea una finestra di dialogo di apertura/creazione di file o cartelle.

```
int  FileSelectDialog(
   string   caption,              // header della finestra
   string   initial_dir,          // directory iniziale
   string   filter,               // filtro di estensione
   uint     flags,                // combinazione di flags
   string&  filenames[],          // array con nomi di file
   string   default_filename      // nome predefinito del file
   );
```

Parametri

caption

[in] Header della finestra di dialogo.

initial_dir

[in] Nome della directory iniziale relativo a MQL5\Files, il cui contenuto deve essere visualizzato nella finestra di dialogo. Se il valore è NULL, MQL5\Files viene visualizzato nella finestra di dialogo.

filter

[in] Filtro di estensione dei file da visualizzare nella finestra di dialogo di selezione. I file di altri formati sono nascosti.

flags

[nel] Combinazione di flags definizione della modalità della finestra di dialogo. I flag sono definiti come segue:
FSD_WRITE_FILE - finestra di dialogo di apertura file;
FSD_SELECT_FOLDER - consente solo la selezione di cartelle;
FSD_ALLOW_MULTISELECT - consente la selezione di più file;
FSD_FILE_MUST_EXIST - i file selezionati dovrebbero esistere;
FSD_COMMON_FOLDER - il file si trova nella cartella comune di tutti i terminali client \Terminal\Common\Files.

filenames[]

[out] Array di stringhe in cui vengono inseriti i nomi dei file/cartelle selezionati.

default_filename

[in] Nome file/cartella predefinito. Se specificato, un nome viene automaticamente aggiunto alla finestra di dialogo aperta e restituito nell'array *filenames[]* durante il test.

Valore di Ritorno

In caso di completamento corretto, la funzione restituisce il numero di file selezionati i cui nomi è possibile ottenerli da *filenames[]*. Se un utente chiude la finestra di dialogo senza selezionare un file, la funzione restituisce 0. In caso di esito negativo, viene restituito un valore inferiore a 0. Il codice di errore può essere ottenuto utilizzando GetLastError().

Nota

Per motivi di sicurezza, lavorare con i file è rigorosamente controllato nel linguaggio MQL5. I file utilizzati nelle operazioni sui file mediante il linguaggio MQL5 non possono trovarsi al di fuori della sandbox dei file (vale a dire, al di fuori della directory MQL5\Files).

Il nome *initial_dir* viene cercato nella directory del terminale client in MQL5\Files (o directory_agente_di_testing\MQL5\Files in caso di testing). Se FSD_COMMON_FOLDER è impostato tra i flag, la ricerca della directory iniziale viene eseguita nella cartella comune di tutti i terminali client \Terminal\Common\Files.

Il parametro *filter* indica file validi e deve essere impostato nel formato "<descrizione 1>|<estensione 1>|<descrizione 2>|<estensione 2>...", ad esempio," File di testo (*.txt)|*.txt|Tutti i file (*.*)|*.*". La prima estensione "File di testo (*.txt)|*.txt" è selezionata come tipo di file predefinito.
Se *filter*=NULL, la maschera di selezione dei file nella finestra di dialogo è "Tutti i file (*.*)|*.*|"

Se il parametro *default_filename* è impostato, chiamando FileSelectDialog() restituisce 1, mentre il valore *default_filename* stesso viene copiato nell'array *filenames[]* durante i testing non-visualizzanti.

La funzione non è consigliata per l'uso in indicatori personalizzati, poiché la chiamata FileSelectDialog() sospende il thread di esecuzione dell'indicatore per tutto il tempo in attesa della risposta dell'utente. Poiché tutti gli indicatori per ciascun simbolo vengono eseguiti in un singolo thread, tutti i charts di tutti i timeframes per questo simbolo vengono sospesi.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione di avvio del programma di Script                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ottiene i nomi dei file di testo per il download dalla cartella comune dei term
   string filenames[];
   if(FileSelectDialog("Selezionare i file da scaricare", NULL,
                       "Text files (*.txt)|*.txt|All files (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames, "data.txt")
     {
      //--- visualizza il nome di ciascun file selezionato
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("File non selezionato");
     }
//---
  }
```

Guarda anche

FileOpen, FileIsExist, FileDelete, FileMove, FolderCreate, FolderDelete, FolderClean, Flags di apertura dei file

FileFindFirst

La funzione inizia la ricerca di file o sottodirectory in una directory in conformità con il filtro specificato.

```
long  FileFindFirst(
   const string   file_filter,          // Stringa - filtro di ricerca
   string&        returned_filename,    // Nome del file o sottodirectory trovata
   int            common_flag=0         // Definisce la ricerca
   );
```

Parametri

file_filter

[in] Filtro di ricerca. Una sottodirectory (o sequenza di sottodirectory nidificate) relativa alla directory \Files, in cui i file devono essere cercati, può essere specificata nel filtro.

returned_filename

[out] Il parametro restituito, dove, in caso di successo, viene piazzato il nome del primo file o sottodirectory trovati. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

common_flag

[in] Flag determinante la posizione del file. Se common_flag = FILE_COMMON, allora il file si trova in una cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, il file si trova in una cartella locale.

Valore restituito

Restituisce l'handle dell'oggetto cercato, che dev'essere usato per l'ulteriore ordinamento dei file e sottodirectory dalla funzione FileFindNext(), o restituisce INVALID_HANDLE quando non ci sono file e sottodirectory corrispondenti al filtro (nel caso particolare - quando la directory è vuota). Dopo aver cercato, l'handle dev'essere chiuso usando la funzione FileFindClose().

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Esempio:

```
//--- display the window of input parameters when launching the script
#property script_show_inputs
//--- filter
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- search for the last backslash
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- the filter contains the folder name
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- get the search handle in the root of the local folder
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- check if the FileFindFirst() is executed successfully
   if(search_handle!=INVALID_HANDLE)
     {
      //--- in a cycle, check if the passed strings are the names of files or directo
      do
        {
         ResetLastError();
         //--- if it's a file, the function returns true, and if it's a directory, it
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- close the search handle
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Vedi anche

FileFindNext, FileFindClose

FileFindNext

La funzione continua la ricerca iniziata da FileFindFirst().

```
bool  FileFindNext(
   long      search_handle,          // Handle di ricerca
   string&   returned_filename       // Nome del file o sottodirectory trovata
   );
```

Parametri

search_handle

[in] Handle di ricerca, recuperato da FileFindFirst().

returned_filename

[out] Il nome del prossimo file o prossima sottodirectory trovata. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

Valore restituito

Se ha successo, restituisce vero, altrimenti false.

Esempio:

```
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- filtro
input string InpFilter="*";
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string file_name;
   int    i=1;
//--- riceve l'handle di ricerca nella root della cartella locale
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- controlla se la funzioneFileFindFirst() è eseguita con successo
   if(search_handle!=INVALID_HANDLE)
     {
      //--- controlla se le stringhe passate nel loop sono file o nomi di directory
      do
        {
         ResetLastError();
         //--- se questo è un file, la funzione restituirà true, se è una directory la
         FileIsExist(file_name);
         PrintFormat("%d : %s nome = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
```

```
      //--- chiude l'handle di ricerca
      FileFindClose(search_handle);
     }
   else
      Print("File non trovato!");
  }
```

Vedi anche

FileFindFirst, FileFindClose

FileFindClose

La funzione chiude l'handle di ricerca.

```
void  FileFindClose(
   long  search_handle      //  Handle di ricerca
   );
```

Parametri

search_handle

[in] Handle di ricerca, recuperato da FileFindFirst().

Valore restituito

Nessun valore restituito.

Nota

La funzione deve essere chiamata per liberare le risorse di sistema.

Esempio:

```
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- filtro
input string InpFilter="*";
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string file_name;
   int    i=1;
//--- riceve l'handle di ricerca nella root della cartella locale
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- controlla se la funzioneFileFindFirst() è eseguita con successo
   if(search_handle!=INVALID_HANDLE)
     {
      //--- controlla se le stringhe passate nel loop sono file o nomi di directory
      do
        {
         ResetLastError();
         //--- se questo è un file, la funzione restituirà true, se è una directory la
         FileIsExist(file_name);
         PrintFormat("%d : %s nome = %s",i,GetLastError()==5018 ? "Directory" : "File"
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- chiude l'handle di ricerca
      FileFindClose(search_handle);
     }
```

```
   else
      Print("File non trovato!");
  }
```

Vedi anche

FileFindFirst, FileFindNext

FileIsExist

Verifica l'esistenza di un file.

```
bool  FileIsExist(
   const string  file_name,        // Nome del file
   int           common_flag=0     // Area di ricerca
   );
```

Parametri

file_name

[in] Il nome del file che viene verificato

common_flag=0

[in] Flag determinante la posizione del file. Se common_flag = FILE_COMMON, allora il file si trova in una cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, il file si trova in una cartella locale.

Valore restituito

Restituisce true, se il file specifico esiste.

Nota

>Il file controllato può rivelarsi essere una sottodirectory. In questo caso, la funzione FileIsExist() restituirà false, mentre l'errore 5018 verrà loggato nella variabile _LastError variable - "Questa è una directory, non un file" (vedi l'esempio per la funzione FileFindFirst).

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Se common_flag = FILE_COMMON, allora la funzione guarda al file in una cartella condivisa per tutti i terminali client \Terminal\Common\Files, altrimenti la funzione cerca il file in una cartella locale (MQL5\Files o MQL5\Tester\Files in caso di testing).

Esempio:

```
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- data per i vecchi file
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string   file_name;      // variabile per memorizzare i nomi dei file
   string   filter="*.txt"; // filtro per cercare i file
   datetime create_date;    // data di creazione file
   string   files[];        // lista dei nomi file
   int      def_size=25;    // grandezza array per default
```

```
   int      size=0;         // numero di file
//--- alloca la memoria per l'array
   ArrayResize(files,def_size);
//--- riceve l'handle di ricerca nella root della cartella locale
   long search_handle=FileFindFirst(filter,file_name);
//--- controlla se FileFindFirst() è stato eseguito con successo
   if(search_handle!=INVALID_HANDLE)
     {
      //--- cerca i file nel loop
      do
        {
         files[size]=file_name;
         //--- incrementa la grandezza dell'array
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- resetta il valore dell'errore
         ResetLastError();
         //--- riceve la data di creazione del file
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- controlla se il file è vecchio
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s file eliminato!",file_name);
            //--- elimina il vecchio file
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- chiude l'handle di ricerca
      FileFindClose(search_handle);
     }
   else
     {
      Print("File non trovato!");
      return;
     }
//--- controlla quale file è rimasto
   PrintFormat("Risultato:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s file esiste!",files[i]);
      else
         PrintFormat("%s file eliminato!",files[i]);
     }
```

```
}
```

Vedi anche

FileFindFirst

FileOpen

La funzione apre il file con il nome e flag specificati.

```
int  FileOpen(
   string  file_name,           // Nome del File
   int     open_flags,          // Combinazione dei flags
   short   delimiter='\t',      // Delimitatore
   uint    codepage=CP_ACP      // Codice della Pagina
   );
```

Parametri

file_name

[in] Il nome del file può contenere sottocartelle. Se il file è aperto per la scrittura, queste sottocartelle verranno create se non ce ne sono.

open_flags

[in] la combinazione di flags determina la modalità di operazione per il file. I flags sono definiti come segue:
FILE_READ il file è aperto per la lettura
FILE_WRITE il file è aperto per la scrittura
FILE_BIN modalità lettura-scrittura binaria (nessuna conversione da string e per string)
FILE_CSV file di tipo csv (tutti gli elementi registrati vengono convertiti in tipo unicode o ansi, e vengono separati da un delimitatore)
FILE_TXT un semplice file di testo (lo stesso del csv, ma un delimitatore non viene preso in considerazione)
FILE_ANSI righe di tipo ANSI (simboli a singolo-byte)
FILE_UNICODE righe di tipo UNICODE (caratteri a doppio-byte)
FILE_SHARE_READ lettura condivisa da diversi programmi
FILE_SHARE_WRITE scrittura condivisa da diversi programmi
FILE_COMMON locazione nel file in una cartella condivisa per tutti i terminali client \Terminal\Common\Files

delimitatore='\t

[in] valore che deve essere usato come un separatore nel file txt o cvs. Se il delimitatore del file csv non viene specificato, il suo default è tab. Se il delimitatore del file txt non viene specificato, allora non viene usato nessun separatore. Se il separatore non è chiaramente impostato a 0, allora non viene usato nessun separatore.

codepage=CP_ACP

[in] Il valore del codice pagina. Per i codici pagina più usati, provvedere appropriate costanti.

Valore restituito

Se un file è stato aperto con successo, la funzione restituisce l'handle file, che è quindi usato per accedere ai dati del file. In caso di fallimento restituisce INVALID_HANDLE.

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Assicurarsi di impostare il flag FILE_ANSI se il file deve essere letto in una codifica specifica (il parametro codepage con il valore una pagina di codice è specificato). Se non esiste alcun flag FILE_ANSI specificato, il file di testo viene letto in Unicode senza alcuna conversione.

Il file è aperto nella cartella del terminale client, nella sottocartella MQL5\files (o testing_agent_directory\MQL5\files in caso di testing). Se FILE_COMMON viene specificato tra i flags, il file viene aperto ina una cartella condivisa per tutti i terminali client MetaTrader 5.

Le "Named pipes" possono essere aperte secondo le seguenti regole:

- Il nome della Pipe è una stringa, che dovrebbe avere il seguente aspetto: "\\servername\pipe\pipename", dove nomeserver - nome del server nella rete, mentre pipeName è un nome di pipe. Se vengono usate le pipes sullo stesso computer, il nome del server può essere omesso ma un punto dev'essere inserito al posto di esso"\\.\pipe\pipename". Un client che cerca di collegasi al pipe dovrebbe conoscere il suo nome.
- FileFlush() e FileSeek() dovrebbero essere chiamate all'inizio di un file tra operazioni sequenziali di lettura della pipe e scrittura ad essa.

Un simbolo speciale '\' viene utilizzato nelle stringhe mostrate. Pertanto, '\' dovrebbe essere raddoppiata quando si scrive un nome in un'applicazione MQL5. Ciò significa che l'esempio di cui sopra dovrebbe avere il seguente aspetto nel codice: "\\\\servername\\pipe\\pipename".

Ulteriori informazioni su come lavorare con le named pipes possono essere trovate nell'articolo "Comunicare con MetaTrader 5 tramite Named Pipe senza utilizzare DLL".

Esempio:

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- metodo scorretto di apertura file
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
   int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
   if(filehandle<0)
     {
      Print("Fallimento nell'apertura del file per percorso assoluto ");
      Print("Error code ",GetLastError());
     }
//--- modalita corretta di lavoro con la "file sandbox"
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("FileOpen OK");
     }
   else Print("Operation FileOpen failed, error ",GetLastError());
//--- altro esempio con la creazione di una directory racchiusa in MQL5\Files\
```

```
   string subfolder="Research";
   filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
      if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("The file most be created in the folder "+terminal_data_path+"\\"+subfolde
     }
   else Print("Apertura file fallita, errore ",GetLastError());
  }
```

Vedi anche

Uso di un Codepage, FileFindFirst, FolderCreate, Flags di apertura file

FileClose

Chiude il file precedentemente aperto da FileOpen().

```
void  FileClose(
   int  file_handle      // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Nessun valore restituito.

Esempio:

```
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri di input
input string InpFileName="file.txt";     // nome del file
input string InpDirectoryName="Data";    // nome della directory
input int    InpEncodingType=FILE_ANSI; // ANSI=32 or UNICODE=64
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- stampa il percorso al file che stiamo per usare
   PrintFormat("Working %s\\Files\\ folder",TerminalInfoString(TERMINAL_DATA_PATH));
//--- resetta il valore dell' errore
   ResetLastError();
//--- apre il file per la lettura (se il file non esiste, avverrà l'errore)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //--- stampa il contenuto del file
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- chiude il file
      FileClose(file_handle);
     }
   else
      PrintFormat("Errore, codice = %d",GetLastError());
  }
```

FileCopy

La funzione copia il file originale da una cartella locale o condivisa in un altro file.

```
bool  FileCopy(
   const string  src_file_name,      // Nome del file sorgente
   int           common_flag,        // Posizione
   const string  dst_file_name,      // Nome del file di destinazione
   int           mode_flags          // Modalità di accesso
   );
```

Parametri

src_file_name

[in] Nome del file da copiare.

common_flag

[in] Flag determinante la posizione del file. Se common_flag = FILE_COMMON, allora il file si trova in una cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, il file si trova in una cartella locale (ad esempio, *common_flag = 0*).

dst_file_name

[in] il nome del file dei risultati.

mode_flags

[in]Flag di Accesso. Il parametro può contenere solo 2 flags: FILE_REWRITE e/o FILE_COMMON - altri flags vengono ignorati. Se il file esiste già, ed il flag FILE_REWRITE non è stato specificato, il file non viene riscritto, e la funzione restituisce false.

Valore restituito

In caso di fallimento la funzione restituisce false.

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Se il nuovo file esiste già, la copia verrà effettuata in base alla disponibilità del flag FILE_REWRITE nel parametro mode_flags.

Esempio:

```
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- parametri di input
input string InpSrc="source.txt";       // sorgente
input string InpDst="destination.txt";  // copia
input int    InpEncodingType=FILE_ANSI; // ANSI=32 or UNICODE=64
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
```

```
voidOnStart()
  {
//--- visualizza il contenuto del sorgente (deve esistere)
   if(!FileDisplay(InpSrc))
      return;
//--- controlla se il file della copia già esiste (può non essere creato)
   if(!FileDisplay(InpDst))
     {
      //--- il file della copia non esiste, copiatura senza il flag FILE_REWRITE (copi
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("Il file è stato copiato!");
      else
         Print("Il file non è stato copiato!");
     }
   else
     {
      //--- il file della copia già esiste, prova copia senza FILE_REWRITE flag (copia
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("Il file è stato copiato!");
      else
         Print("Il file non è stato copiato!");
      //--- il contenuto file InpDst rimane lo stesso
      FileDisplay(InpDst);
      //--- copia ancora una volta con il flag FILE_REWRITE (copiatura corretta se il
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("Il file è stato copiato!");
      else
         Print("Il file non è stato copiato!");
     }
//--- riceve la copia del file InpSrc
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------------------+
//| Legge il contenuto del file                                                  |
//+------------------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- resetta il valore dell' errore
   ResetLastError();
//--- apre il file
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- visualizza il contenuto del file nel loop
      Print("+----------------------+");
      PrintFormat("Nome del file = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
```

```
      //--- chiude il file
      FileClose(file_handle);
      return(true);
     }
//--- fallimento nell'aprire il file
   PrintFormat("%s non è aperto, errore = %d",file_name,GetLastError());
   return(false);
  }
```

FileDelete

Elimina il file specificato in una cartella locale del terminale cliente.

```
bool  FileDelete(
   const string  file_name,     // Nome del file da eliminare
   int           common_flag=0  // Posizione del file da eliminare
   );
```

Parametri

file_name

[in] Nome file.

common_flag=0

[in] Flag determinante la posizione. Se common_flag = FILE_COMMON, allora il file si trova in una cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, il file si trova in una cartella locale.

Valore restituito

In caso di fallimento la funzione restituisce false.

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Elimina il file specificato da una cartella locale del terminale client (MQL5\files oppure MQL5\tester\files in caso di testing). Se common_flag = FILE_COMMON, allora la funzione rimuove il file dalla cartella condivisa per tutti i terminali client.

Esempio:

```
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- data per i vecchi file
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string   file_name;      // variabile per memorizzare i nomi dei file
   string   filter="*.txt"; // filtro per cercare i file
   datetime create_date;    // data di creazione file
   string   files[];        // lista dei nomi file
   int      def_size=25;    // grandezza array per default
   int      size=0;         // numero di file
//--- alloca la memoria per l'array
   ArrayResize(files,def_size);
//--- riceve l'handle di ricerca nella root della cartella locale
```

```
   long search_handle=FileFindFirst(filter,file_name);
//--- controlla se FileFindFirst() è stato eseguito con successo
   if(search_handle!=INVALID_HANDLE)
     {
      //--- cerca i file nel loop
      do
        {
         files[size]=file_name;
         //--- incrementa la grandezza dell'array
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- resetta il valore dell'errore
         ResetLastError();
         //--- riceve la data di creazione del file
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- controlla se il file è vecchio
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s file eliminato!",file_name);
            //--- elimina il vecchio file
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- chiude l'handle di ricerca
      FileFindClose(search_handle);
     }
   else
     {
      Print("File non trovato!");
      return;
     }
//--- controlla quale file è rimasto
   PrintFormat("Risultato:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s file esiste!",files[i]);
      else
         PrintFormat("%s file eliminato!",files[i]);
     }
  }
```

FileMove

Sposta il file da una cartella locale o condivisa in un altra cartella.

```
bool  FileMove(
   const string  src_file_name,     // Il nome del file per l'operazione di spostamento
   int           common_flag,       // Locazione
   const string  dst_file_name,     // Nome del file di destinazione
   int           mode_flags         // Modalità di accesso
   );
```

Parametri

src_file_name

[in] Nome del file da spostare/rinominare.

common_flag

[in] Flag determinante la posizione del file. Se common_flag = FILE_COMMON, allora il file si trova in una cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, il file si trova in una cartella locale (*common_flag=0*).

dst_file_name

[in] Nome del file dopo l'operazione

mode_flags

[in]Flag di Accesso. Il parametro può contenere solo 2 flags: FILE_REWRITE e/o FILE_COMMON - altri flags vengono ignorati. Se il file già esiste ed il flag FILE_REWRITE non è specificato, il file non verrà riscritto, e la funzione resituità false.

Valore restituito

In caso di fallimento la funzione restituisce false.

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Se il nuovo file esiste già, la copia verrà effettuata in base alla disponibilità del flag FILE_REWRITE nel parametro mode_flags.

Esempio:

```
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- parametri di input
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
```

```
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- riceve i percorsi dei file
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- controlla se il file sorgente esiste (se no - esce)
   if(FileIsExist(src_path))
      PrintFormat("il file %s esiste nella cartella %s\\Files\\%s",InpSrcName,local,In
   else
     {
      PrintFormat("Errore, %s file sorgente non trovato",InpSrcName);
      return;
     }
//--- controlla se il file risultato già esiste
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("il file %s esiste nella cartella %s\\Files\\%s",InpDstName,common,I
      //--- il file esiste, lo spostamento dev'essere eseguito con la flag FILE_REWRI
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("il %s è stato spostato",InpSrcName);
      else
         PrintFormat("Errore! Code = %d",GetLastError());
     }
   else
     {
      PrintFormat("il file %s non esiste nella cartella %s\\Files\\%s",InpDstName,comm
      //--- il file non esiste, lo spostamento dev'essere eseguito senza la flag FILE_
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("il %s è stato spostato",InpSrcName);
      else
         PrintFormat("Errore! Code = %d",GetLastError());
     }
//--- il file è stato spostato; controlliamo
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Ok!");
   else
      Print("Errore!");
  }
```

Vedi anche

FileIsExist

FileFlush

Scrive su un disco tutti i dati rimasti nel buffer input/output del file.

```
void  FileFlush(
   int  file_handle      // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Nessun valore restituito.

Nota

Quando si scrive un file, i dati possono essere effettivamente trovato lì solo dopo qualche tempo. Per salvare i dati nel file immediatamente, utilizzare la funzione FileFlush(). Se la funzione non viene utilizzata, parte dei dati che non sono stati archiviati nel disco ancora, verranno forzatamente scritta lì solo quando il file viene chiuso mediante la funzione FileClose().

La funzione deve essere utilizzata quando i dati scritti sono di un certo valore. Va tenuto presente che chiamate di funzioni frequenti possono influire sulla velocità di funzionamento del programma.

La funzione FileFlush() deve essere chiamata tra le operazioni di lettura da un file, e di scrittura su esso.

Esempio:

```
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- nome del file per la scrittura
input string InpFileName="example.csv"; // nome del file
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- resetta il valore dell'errore
   ResetLastError();
//--- apre il file
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- scrive i dati sul file
      for(int i=0;i<1000;i++)
        {
         //--- chiama la funzione di scrittura
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sy
         //--- salva i dati sul disco ogni 128essima iterazione
```

```
         if((i & 127)==127)
           {
            //--- ora i dati verranno piazzati nel file e non verranno persi in caso c
            FileFlush(file_handle);
            PrintFormat("i = %d, OK",i);
           }
         //--- 0.01 secondi di pausa
         Sleep(10);
        }
      //--- chiude il file
      FileClose(file_handle);
     }
   else
      PrintFormat("Errore, codice = %d",GetLastError());
  }
```

Vedi anche

FileClose

FileGetInteger

Ottiene una proprietà integer di un file. Ci sono due varianti della funzione.

1. Ottine una proprietà dall'handle del file.

```
long  FileGetInteger(
   int                          file_handle,          // File handle
   ENUM_FILE_PROPERTY_INTEGER   property_id           // ID proprietà
   );
```

2. Ottiene la proprietà dal nome del file.

```
long  FileGetInteger(
   const string                 file_name,            // Nome File
   ENUM_FILE_PROPERTY_INTEGER   property_id,          // ID proprietà
   bool                         common_folder=false   // Il file è visualizzato in una
   );                                                 // o in una cartella comune a tut
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

file_name

[in] Nome file.

property_id

[in] ID proprietà File . Il valore può essere uno dei valori dell'enumerazione ENUM_FILE_PROPERTY_INTEGER. Se la seconda variante della fuzione viene usata, puoi ricevere solo i valori delle seguenti proprietà: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE e FILE_SIZE.

common_folder=false

[in] Punta alla locazione del file. Se il parametro è false, viene visualizzata la cartella dati del terminale. Altrimenti è sottointeso che il file è in una cartella condivisa da tutti i terminali \Terminal\Common\Files (FILE_COMMON).

Valore restituito

Il valore della proprietà. In caso di errore, viene restituito -1. Per ottenere un codice di errore usare la funzione GetLastError().>

Se una cartella viene specificata quando si ottengono le proprietà dal nome, la funzione avrà errore 5018 (ERR_MQL_FILE_IS_DIRECTORY) in ogni caso, though the return value will be correct.

Nota

La funzione cambia sempre il codice dell'errore. In caso di completamento di successo, il codice dell'errore è resettato a NULL.

Esempio:

```
//--- mostra la finestra dei parametri di input quando lancia lo script
```

```
#property script_show_inputs
//--- parametri di input
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- apre il file
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- fa il print di tutte le informazioni sul file
      Print(InpFileName," file info:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"date");
      FileInfo(handle,FILE_MODIFY_DATE,l,"date");
      FileInfo(handle,FILE_ACCESS_DATE,l,"date");
      FileInfo(handle,FILE_SIZE,l,"other");
      FileInfo(handle,FILE_POSITION,l,"other");
      FileInfo(handle,FILE_END,l,"bool");
      FileInfo(handle,FILE_IS_COMMON,l,"bool");
      FileInfo(handle,FILE_IS_TEXT,l,"bool");
      FileInfo(handle,FILE_IS_BINARY,l,"bool");
      FileInfo(handle,FILE_IS_CSV,l,"bool");
      FileInfo(handle,FILE_IS_ANSI,l,"bool");
      FileInfo(handle,FILE_IS_READABLE,l,"bool");
      FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
      //--- chiude il file
      FileClose(handle);
     }
   else
      PrintFormat("%s il file non è stato aperto, ErrorCode = %d",InpFileName,GetLastE
  }
//+------------------------------------------------------------------------------+
//| Mostra il valore della proprietà del file                                    |
//+------------------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- ricevere il valore della proprietà
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- il valore ricevuto, lo mostra nel formato corretto
```

```
      if(!StringCompare(type,"bool"))
         Print(EnumToString(id)," = ",l ? "true" : "false");
      if(!StringCompare(type,"date"))
         Print(EnumToString(id)," = ",(datetime)l);
      if(!StringCompare(type,"other"))
         Print(EnumToString(id)," = ",l);
     }
   else
      Print("Error, Code = ",GetLastError());
  }
```

Vedi anche

Operazioni sui File, Proprietà dei File

FileIsEnding

Definisce la fine di un file nel processo di lettura.

```
bool  FileIsEnding(
   int  file_handle      // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

La funzione restituisce true se la fine del file è stata raggiunta nel processo di lettura o spostamento del puntatore file.

Nota

Per definire la fine del file, la funzione tenta di leggere la successiva stringa da esso. Se la stringa non esiste, la funzione restituisce true, altrimenti restituisce false.

Esempio:

```
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri di input
input string InpFileName="file.txt";     // nome del file
input string InpDirectoryName="Data";    // nome della directory
input int    InpEncodingType=FILE_ANSI; // ANSI=32 or UNICODE=64
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- stampa il percorso al file che stiamo per usare
   PrintFormat("Working %s\\Files\\ folder",TerminalInfoString(TERMINAL_DATA_PATH));
//--- resetta il valore dell' errore
   ResetLastError();
//--- apre il file per la lettura (se il file non esiste, avverrà l'errore)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //--- stampa il contenuto del file
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- chiude il file
      FileClose(file_handle);
     }
   else
      PrintFormat("Errore, codice = %d",GetLastError());
```

```
}
```

FileIsLineEndin

Definisce la fine della riga in un file di testo nel processo di lettura.

```
bool  FileIsLineEnding(
   int  file_handle      // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Restituisce true se il processo di lettura del file txt o csv raggiunge la fine della riga (i caratteri CR-LF).

Example (il file ottenuto durante l'esecuzione di un esempio per la funzione FileWriteString viene qui utilizzato)

```
//+------------------------------------------------------------------------------+
//|                                                  Demo_FileIsLineEnding.mq5   |
//|                                Copyright 2013, MetaQuotes Software Corp.     |
//|                                              https://www.mql5.com            |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parametri per la lettura dei dati
input string InpFileName="RSI.csv";   // nome del file
input string InpDirectoryName="Data"; // nome della directory
//--- buffers indicatore
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- variabili overbought
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
```

```
//--- variabili oversold
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- variabili di grandezze array per default
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- alloca la memoria per gli array
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      double value;
      //--- legge i dati dal file
      while(!FileIsEnding(file_handle))
        {
         //--- legge i primi valori nella stringa
         value=FileReadNumber(file_handle);
         //--- legge da diversi arry secondo il risultato della funzione
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- lega gli array
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
```

```
//---- imposta i valori dell'indicatore che non saranno visibili sul chart
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Legge i dati stringa del file                                                |
//+------------------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- legge fino alla che viene raggiunta la fine della stringa o del file
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- slitta il carriage dopo aver letto il numero
      if(flag)
         FileReadNumber(file_handle);
      //--- memorizza la data corrente
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- incrementa la grandezza dell'array se necessario
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- va oltre la prima iterazione
      flag=true;
     }
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
```

```
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- controlla se qualunque data è ancora presente
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- se le date coincidono, la barra è in area overbought
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - color rosso
               color_buff[i]=0;
               //--- incrementa il contatore
               ovb_ind=j+1;
               break;
              }
           }
      //--- controlla se esistono ancora dati
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- se le date coincidono, la barra è in area oversold
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - colore blu
               color_buff[i]=1;
               //--- incrementa il contatore
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

```
//+------------------------------------------------------------------+
//| Event handler ChartEvent                                         |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- cambia lo spessore indicatore secondo la scala
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Vedi anche

FileWriteString

FileReadArray

Legge da un file di tipo BIN l'array di qualunque tipo eccetto stringa (può essere un array di strutture, non contenente stringhe, ed array dinamici).

```
uint  FileReadArray(
   int     file_handle,               // File handle
   void&   array[],                   // Array da registrare
   int     start=0,                   // posizione di inizio dell'array, da scrivere
   int     count=WHOLE_ARRAY          // conteggio a leggere
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

array[]

[out] Un array dove i dati verranno caricati.

start=0

[in] Posizione di inizio per scrivere nell'array.

count=WHOLE_ARRAY

[in] Numero di elementi da leggere. Per default, legge l'intero array (count=WHOLE_ARRAY).

Valore restituito

Numero di elementi letti.

Nota

Array stringa può essere sola lettura dal file di tipo TXT. Se necessario, la funzione prova ad incrementare la grandezza dell'array.

Esempio (il file ottenuto dopo l'esecuzione dell'esempio per la funzione FileWriteArray è qui usato)

```
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- parametri di input
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------------------+
//| Struttura per memorizzare i dati dei prezzi                                  |
//+------------------------------------------------------------------------------+
struct prices
  {
   datetime          date; // date
   double            bid;  // prezzo bid
   double            ask;  // prezzo ask
  };
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
```

```
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
//--- struttura array
   prices arr[];
//--- percorso file
   string path=InpDirectoryName+"//"+InpFileName;
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- legge tutti i dati dal file nell'array
      FileReadArray(file_handle,arr);
      //--- riceve la grandezza dell'array
      int size=ArraySize(arr);
      //--- fa il print dei dati dall'array
      for(int i=0;i<size;i++)
         Print("Date = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Total data = ",size);
      //--- chiude il file
      FileClose(file_handle);
     }
   else
      Print("Fallimento nell'aprire il file, errore ",GetLastError());
  }
```

Vedi anche

Variabili, FileWriteArray

FileReadBool

Legge dal file di tipo CSV la stringa dalla posizione corrente fino al delimitatore (o fino alla fine della riga del testo) e converte la stringa letta in valore di tipo bool.

```
bool  FileReadBool(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

La riga letta può essere impostata a "true", "false" o la simbolica rappresentazione di integers "0" o "1". Un valore non-zero viene convertito nel valore logico true. La funzione restituisce il valore convertito.

Esempio (il file ottenuto fopo l'esecuzione dell'esempio per la funzione FileWrite è qui utilizzato)

```
//+------------------------------------------------------------------------------------+
//|                                                          Demo_FileReadBool.mq5 |
//|                                       Copyright 2013, MetaQuotes Software Corp. |
//|                                                          https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- parametri per la lettura dei dati
input string InpFileName="MACD.csv";  // nome del file
input string InpDirectoryName="Data"; // nome della directory
//--- variabili globali
int      ind=0;       // indice
double   upbuff[];    // buffer indicatore di freccia su
```

```
double   downbuff[];  // buffer indicatore di freccia giu
bool     sign_buff[]; // array signal (true - buy, false - sell)
datetime time_buff[]; // array dell'orario di arrivo di signal
int      size=0;      // grandezza dell'array signal
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s è aperto per la lettura",InpFileName);
      //--- prima, legge il numero di segnali
      size=(int)FileReadNumber(file_handle);
      //--- alloca la memoria per gli array
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- legge i dati dal file
      for(int i=0;i<size;i++)
        {
         //--- orario di signal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- valore di signal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- chiude il file
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- lega gli array
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- imposta il codice del simbolo per il disegno in PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- imposta i valori dell'indicatore che non verranno mostrati sul chart
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione di iterazione indicatore personalizato                          |
//+--------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      upbuff[i]=0;
      downbuff[i]=0;
      //--- controlla se ci sono ancora dati presenti
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se le date coincidono, usa il valore dal file
            if(time[i]==time_buff[j])
              {
               //--- disegna la freccia secondo il segnale
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- incrementa il contatore
               ind=j+1;
               break;
              }
           }
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Tipo bool, FileWrite

FileReadDatetime

Legge dal file di tipo CSV una stringa di uno dei formati : "AAAA.MM.GG OO:MI:SS", "AAAA.MM.GG" or "OO:MI:SS" - e la converte in un valore di tipo datetime.

```
datetime  FileReadDatetime(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Il valore di tipo datetime.

Esempio (il file ottenuto fopo l'esecuzione dell'esempio per la funzione FileWrite è qui utilizzato)

```
//+------------------------------------------------------------------------------+
//|                                                 Demo_FileReadDateTime.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- parametri per la lettura dei dati
input string InpFileName="MACD.csv";  // nome del file
input string InpDirectoryName="Data"; // nome della directory
//--- variabili globali
int      ind=0;       // indice
double   upbuff[];    // buffer indicatore di freccia su
double   downbuff[];  // buffer indicatore di freccia giu
bool     sign_buff[]; // array signal (true - buy, false - sell)
```

```
datetime time_buff[]; // array dell'orario di arrivo di signal
int      size=0;      // grandezza dell'array signal
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s è aperto per la lettura",InpFileName);
      //--- prima, legge il numero di segnali
      size=(int)FileReadNumber(file_handle);
      //--- alloca la memoria per gli array
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- legge i dati dal file
      for(int i=0;i<size;i++)
        {
         //--- orario di signal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- valore di signal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- chiude il file
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- lega gli array
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- imposta il codice del simbolo per il disegno in PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- imposta i valori dell'indicatore che non verranno mostrati sul chart
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
```

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      upbuff[i]=0;
      downbuff[i]=0;
      //--- controlla se ci sono ancora dati presenti
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se le date coincidono, usa il valore dal file
            if(time[i]==time_buff[j])
              {
               //--- disegna la freccia secondo il segnale
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- incrementa il contatore
               ind=j+1;
               break;
              }
           }
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Tipo datetime, StringToTime, TimeToString, FileWrite

FileReadDouble

Legge un numero a doppia-precisione floating point (double) dalla corrente posizione del file binario.

```
double  FileReadDouble(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Il valore di tipo double.

Nota

Per più dettagli riguardo l'errore, chiamare GetLastError().

Esempio (il file ottenuto dopo l'esecuzione dell'esempio per la funzione FileWriteDouble è qui utilizzato)

```
//+------------------------------------------------------------------------------------+
//|                                                        Demo_FileReadDouble.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                                             https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- parametri di lettura dei dati
input string InpFileName="MA.csv";     // nome del file
input string InpDirectoryName="Data"; // nome della directory
//--- variabili globali
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- buffer indicatore
double   buff[];
```

```
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- prima, legge la quantità di dati nel file
      size=(int)FileReadDouble(file_handle);
      //--- alloca la memoria per gli array
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- legge i dati dal file
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- lega l'array al buffer indicatore con l'indice 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- imposta i valori dell'indicatore che non saranno visibili sul chart
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      buff[i]=0;
      //--- controlla se esistono ancora dati
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se le date coincidono, viene usato il valore dal file
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //--- incrementa il contatore
               ind=j+1;
               break;
              }
           }
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Tipi Real (double, float), StringToDouble, DoubleToString, FileWriteDouble

FileReadFloat

Legge un numero a singola-precisione floating point (float) dalla corrente posizione del file binario.

```
float  FileReadFloat(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Il valore di tipo float.

Nota

Per più dettagli riguardo l'errore, chiamare GetLastError().

Esempio (il file ottenuto dopo l'esecuzione dell'esempio per la funzione FileWriteFloat è qui usato)

```
//+------------------------------------------------------------------------------+
//|                                                    Demo_FileReadFloat.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parametri per la lettura dei dati
input string InpFileName="Close.bin"; // nome del file
input string InpDirectoryName="Data"; // nome della directory
//--- variabili globali
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- buffers indicatore
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------------------+
```

```
//| Funzione di inizializzazione Indicatore Personalizzato                   |
//+------------------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- alloca la memoria per gli array
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- legge i dati dal file
      while(!FileIsEnding(file_handle))
        {
         //--- legge i valori di tempo e di prezzo
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- incrementa la grandezza dell'array se non viene riempito
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono letti, il file %s è chiuso",InpFileName);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- lega gli array ai buffer indicatore
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- imposta i valori dell'indicatore che non saranno visibili sul chart
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                          |
```

```
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      buff[i]=0;
      color_buff[i]=0; // colore rosso per default
      //--- controlla se ci sono ancora dati presenti
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se le date coincidono, viene usato il valore dal file
            if(time[i]==time_buff[j])
              {
               //--- riceve il prezzo
               buff[i]=close_buff[j];
               //--- se il prezzo corrente eccede quello previsto, il colore è blu
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- incrementa il contatore
               ind=j+1;
               break;
              }
           }
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Tipi reali (double, float), FileReadDouble, FileWriteFloat

FileReadInteger

La funzione legge int, short o il valore char dalla posizione corrente del puntatore del file a seconda della lunghezza specificata in byte.

```
int  FileReadInteger(
   int  file_handle,         // File handle
   int  size=INT_VALUE       // Grandezza di un integer in byte
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

size=INT_VALUE

[in] Numero di byte (fino a 4 inclusivi), che dev'essere letto. Le corrispondenti costanti vengono fornite: CHAR_VALUE = 1, SHORT_VALUE = 2 and INT_VALUE = 4, così la funzione può leggere il valore intero del tipo char, short o int.

Valore restituito

Un valore del tipo int. Il risultato di questa funzione deve essere esplicitamente castato ad un tipo target, cioè al tipo di dati che si ha bisogno di leggere. Giacchè viene restituito un valore di tipo int, esso può essere facilmente convertito in qualunque valore integer. Il puntatore del file è slittato per il numero dei byte letti.

Nota

Quando si legge meno di 4 byte, il risultato ricevuto è sempre positivo. If one or two bytes are read, the sign of the number can be determined by explicit casting to type char (1 byte) or short (2 bytes). Ottenere il segno per un numero tre-byte non è banale, poiché non vi è corrispondenza di tipo sottostante.

Esempio (il file ottenuto fopo l'esecuzione dell'esempio per la funzione FileWriteInteger è qui utilizzato)

```
//+------------------------------------------------------------------------------+
//|                                                  Demo_FileReadInteger.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri per la lettura dei dati
input string InpFileName="Trend.bin"; // nome del file
input string InpDirectoryName="Data"; // nome della directory
//--- variabili globali
int      ind=0;
int      size=0;
datetime time_buff[];
//--- buffers indicatore
double   buff[];
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- alloca la memoria per l'array
   ArrayResize(time_buff,def_size);
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- variabili aggiuntive
      int    arr_size;
      uchar  arr[];
      //--- legge i dati dal file
      while(!FileIsEnding(file_handle))
        {
         //--- trova come molti simboli vengono usati per scrivere l'orario
         arr_size=FileReadInteger(file_handle,INT_VALUE);
         ArrayResize(arr,arr_size);
         for(int i=0;i<arr_size;i++)
            arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
         //--- conserva i valori temporali
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- incrementa le grandezze degli array se non vengono riempiti
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
        }
      //--- chiude il file
      FileClose(file_handle);
```

```
      PrintFormat("I dati vengono letti, il file %s è chiuso",InpFileName);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- lega l'array al buffer indicatore
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- imposta i valori dell'indicatore che non saranno visibili sul chart
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(close,false);
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      buff[i]=0;
      //--- controlla se ci sono ancora dati presenti
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se le date coincidono, viene usato il valore dal file
            if(time[i]==time_buff[j])
              {
               //--- riceve il prezzo
               buff[i]=close[i];
               //--- incrementa il contatore
               ind=j+1;
               break;
              }
```

```
            }
         }
      }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

IntegerToString, StringToInteger, Integer types, FileWriteInteger

FileReadLong

La funzione legge un numero intero di tipo long (8 byte) dalla posizione corrente del file binario.

```
long  FileReadLong(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Il valore di tipo long.

Esempio (il file ottenuto durante l'esecuzione di un esempio per la funzione FileWriteLong viene qui utilizzato)

```
//+------------------------------------------------------------------------------+
//|                                                      Demo_FileReadLong.mq5 |
//|                                  Copyright 2013, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- parametri per la lettura dei dati
input string InpFileName="Volume.bin"; // nome del file
input string InpDirectoryName="Data";  // nome directory
//--- variabili globali
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- buffers indicatore
double   buff[];
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
```

```
  {
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s viene aperto per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- prima, legge la quantità di dati nel file
      size=(int)FileReadLong(file_handle);
      //--- alloca la memoria per gli array
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- legge i dati dal file
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono letti, il file %s è chiuso",InpFileName);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- associa l'array al buffer indicatore con indice 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
/ ---- imposta i valori degli indicatori che saranno visibili sul grafico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

```
   ArraySetAsSeries(time,false);
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      buff[i]=0;
      //--- controlla se ci sono ancora dati presenti
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se le date coincidono, viene usato il valore dal file
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Integer types, FileReadInteger, FileWriteLong

FileReadNumber

La funzione legge dal file CSV una stringa dalla posizione corrente fino ad un separatore (o fino alla fine di una stringa di testo) e converte la stringa letta in un valore di tipo double.

```
double  FileReadNumber(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Il valore di tipo double.

Example (il file ottenuto durante l'esecuzione di un esempio per la funzione FileWriteString viene qui utilizzato)

```
//+------------------------------------------------------------------------------+
//|                                                    Demo_FileReadNumber.mq5 |
//|                                 Copyright 2013, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parametri per la lettura dei dati
input string InpFileName="RSI.csv";   // nome del file
input string InpDirectoryName="Data"; // nome della directory
//--- buffers indicatore
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- variabili overbought
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
```

```
//--- variabili oversold
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- variabili di grandezze array per default
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- alloca la memoria per gli array
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      double value;
      //--- legge i dati dal file
      while(!FileIsEnding(file_handle))
        {
         //--- legge i primi valori nella stringa
         value=FileReadNumber(file_handle);
         //--- legge da diversi arry secondo il risultato della funzione
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- lega gli array
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
```

```
//---- imposta i valori dell'indicatore che non saranno visibili sul chart
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Legge i dati stringa del file                                                |
//+------------------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- legge fino alla che viene raggiunta la fine della stringa o del file
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- slitta il carriage dopo aver letto il numero
      if(flag)
         FileReadNumber(file_handle);
      //--- memorizza la data corrente
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- incrementa la grandezza dell'array se necessario
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- va oltre la prima iterazione
      flag=true;
     }
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
```

```
//--- il loop per la barra che non è stata ancora maneggiata
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- controlla se qualunque data è ancora presente
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- se le date coincidono, la barra è in area overbought
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - color rosso
               color_buff[i]=0;
               //--- incrementa il contatore
               ovb_ind=j+1;
               break;
              }
           }
      //--- controlla se esistono ancora dati
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- se le date coincidono, la barra è in area oversold
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - colore blu
               color_buff[i]=1;
               //--- incrementa il contatore
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

```
//+------------------------------------------------------------------+
//| Event handler ChartEvent                                         |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- cambia lo spessore indicatore secondo la scala
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Vedi anche

FileWriteString

FileReadString

La funzione legge da una stringa dalla posizione corrente di un puntatore di file in un file.

```
string  FileReadString(
   int  file_handle,     // File handle
   int  length=-1        // Lunghezza della stringa
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

length=-1

[in] Numero di caratteri da leggere.

Valore restituito

Linea letta (stringa).

Nota

Durante si legge da un file-bin. la lunghezza della stringa da leggere deve essere specificata. Quando si legge da un file-txt la lunghezza della stringa non è richiesta, e la stringa sarà letta dalla posizione attuale al carattere di avanzamento riga "\r\n". Durante la lettura da un file CSV, la lunghezza della stringa non è necessaria inoltre, la stringa verrà letta dalla posizione corrente fino al più vicino delimitatore o fino al carattere di fine stringa di testo.

Se il file viene aperto con il flag FILE_ANSI, allora la riga letta viene convertita in Unicode.

Esempio (il file ottenuto fopo l'esecuzione dell'esempio per la funzione FileWriteInteger è qui utilizzato)

```
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- parametri per la lettura dei dati
input string InpFileName="Trend.bin"; // nome del file
input string InpDirectoryName="Data"; // nome della directory
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- variabili aggiuntive
      int    str_size;
```

```
      string str;
      //--- legge i dati dal file
      while(!FileIsEnding(file_handle))
        {
         //--- trova come molti simboli vengono usati per scrivere l'orario
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- legge la stringa
         str=FileReadString(file_handle,str_size);
         //--- fa il print della stringa
         PrintFormat(str);
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono letti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

Tipo string, Conversione Dati, FileWriteInteger

FileReadStruct

La funzione legge il contenuto in una struttura passata come parametro da un file binario, a partire dalla posizione corrente del puntatore del file.

```
uint  FileReadStruct(
   int           file_handle,        // File handle
   const void&   struct_object,      // struttura target per il quale il contenuto vier
   int           size=-1             // grandezza struttura in bytes
   );
```

Parametri

file_handle

[in] Descrittore di un file-bin aperto.

struct_object

[out] L'oggetto di questa struttura. La struttura non deve contenere stringhe, array dinamici o funzioni virtuali.

size=-1

[in] Numero di byte che devono essere letti. Se la grandezza non è specificata o il valore indicato è maggiore della grandezza della struttura, viene utilizzata la grandezza esatta della struttura specificata.

Valore restituito

In caso di successo la funzione restituisce il numero di byte letti. Il puntatore del file viene spostato per lo stesso numero di byte.

Esempio (il file ottenuto dopo aver usato l'esempio per la funzione for FileWriteStruct è qui utilizzato)

```
//+------------------------------------------------------------------------------+
//|                                                    Demo_FileReadStruct.mq5 |
//|                                 Copyright 2013, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Candele"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- parametri per la ricezione dei dati
input string  InpFileName="EURUSD.txt"; // nome del file
```

```
input string  InpDirectoryName="Data";  // nome directory
//+------------------------------------------------------------------------------+
//| Struttura per memorizzare i dati della candela                               |
//+------------------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // prezzo apertura
   double            close; // prezzo chiusura
   double            high;  // prezzo più alto
   double            low;   // prezzo più basso
   datetime          date;  // data
  };
//--- buffers indicatore
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- variabili globali
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- apre il file
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_
      //--- legge i dati dal file
      while(!FileIsEnding(file_handle))
        {
         //--- scrive i dati sull' array
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
         //--- controlla se l'array è stato riempito
         if(size==default_size)
           {
            //--- incrementa le dimensioni dell'array
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
```

```
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono letti, il file %s è chiuso",InpFileName);
     }
   else
     {
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
      return(INIT_FAILED);
     }
//--- mappatura buffers indicatore
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- valore vuoto
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- il ciclo per le candele che non sono state ancora trattate
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 per default
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //--- controlla se ci sono ancora dati presenti
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se le date coincidono, viene usato il valore dal file
            if(time[i]==cand_buff[j].date)
```

```
               {
                open_buff[i]=cand_buff[j].open;
                close_buff[i]=cand_buff[j].close;
                high_buff[i]=cand_buff[j].high;
                low_buff[i]=cand_buff[j].low;
                //--- incrementa il contatore
                ind=j+1;
                break;
               }
            }
         }
      }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Strutture e Classi, FileWriteStruct

FileSeek

La funzione sposta la posizione del puntatore di file per un determinato numero di byte rispetto alla posizione specificata.

```
bool  FileSeek(
   int                  file_handle,     // File handle
   long                 offset,          // In byte
   ENUM_FILE_POSITION   origin           // Posizione per riferimento
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

offset

[in] Lo slttamento in byte (può assumere un valore negativo).

origin

[in] Il punto di partenza per lo spostamento. Può essere uno dei valori di ENUM_FILE_POSITION.

Valore restituito

In caso di successo la funzione restituisce true, altrimenti false. Per ottenere informazioni sull' errore chiamare la funzione GetLastError().

Nota

Se l'esecuzione della funzione FileSeek() risulta in uno slittamento negativo (va oltre il "livello di confine" del file), il puntatore del file viene impostato all'inizio del file.

Se una posizione viene impostata al di là del "limite destro" del file (più grande della dimensione del file), la scritta accanto al file non verrà effettuata a partire dalla fine del file, ma dalla posizione impostata. In questo caso, valori indefiniti verranno scritti per la fine del file precedente e la posizione impostata.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_FileSeek.mq5 |
//|                                  Copyright 2013, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- parametri di input
input string InpFileName="file.txt";     // nome del file
input string InpDirectoryName="Data";    // nome della directory
input int    InpEncodingType=FILE_ANSI; // ANSI=32 or UNICODE=64
```

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- specifica il valore della variabile per generare numeri casuali
   _RandomSeed=GetTickCount();
//--- variabili per le posizioni dei punti di inizio della stringa
   ulong pos[];
   int   size;
//--- resetta il valore dell' errore
   ResetLastError();
//--- apre il file
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      //--- riceve la posizione d'inizio per ogni stringa nel file
      GetStringPositions(file_handle,pos);
      //--- definisce il numero di stringhe nel file
      size=ArraySize(pos);
      if(!size)
        {
         //--- si ferma se il file non ha stringhe
         PrintFormat("il file %s è vuoto!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- fa una selezione casuale di un numero di stringa
      int ind=MathRand()%size;
      //--- slitta la posizione al punto iniziale della stringa
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- legge e fa il print della stringa con il numero ind
         PrintFormat("Testo stringa con numero %d : \"%s\"",ind,FileReadString(file_ha
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("il file %s è stato chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
//+------------------------------------------------------------------------------+
//| La funzione definisce i punti di inizio per ogni stringa nel file e          |
//| li piazza nell'array arr                                                     |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
```

```
//--- grandezza array di default
   int def_size=127;
//--- alloca la memoria per l'array
   ArrayResize(arr,def_size);
//--- contatore stringa
   int i=0;
//--- se questa non è la fine del file, allora c'è ancora una stringa
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // il file è vuoto, esce
//--- definisce lo slittamento in byte a seconda della codifica
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- va attraverso le stringhe nel loop
   while(1)
     {
      //--- legge la stringa
      FileReadString(handle);
      //--- controlla la fine del file
      if(!FileIsEnding(handle))
        {
         //--- memorizza la posizione della prossima stringa
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- incrementa la grandezza dell'array se è stato riempito
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // fine del file, uscita
     }
//--- definisce l'attuale grandezza dell'array
   ArrayResize(arr,i);
  }
```

FileSize

La funzione restituisce la dimensione del file in byte.

```
ulong  FileSize(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Il valore di tipo int.

Nota

Per ottenere informazioni sull' errore chiamare GetLastError().

Esempio:

```
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri di input
input ulong  InpThresholdSize=20;        // soglia della grandezza del file in kiloby
input string InpBigFolderName="big";     // cartella per file larghi
input string InpSmallFolderName="small"; // cartella per file piccoli
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string   file_name;      // variabile per memorizzare i nomi dei file
   string   filter="*.csv"; // filtro per la ricerca dei file
   ulong    file_size=0;    // grandezza del file in byte
   int      size=0;         // numero di file
//--- fa il print del percorso del file con cui stiamo per lavorare
   PrintFormat("Working in %s\\Files\\ folder",TerminalInfoString(TERMINAL_COMMONDATA_
//--- riceve l'handle di ricerca nella root della cartella comune a tutti i terminali
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- controlla se FileFindFirst() è stato eseguito con successo
   if(search_handle!=INVALID_HANDLE)
     {
      //--- sposta i file nel ciclo secondo la loro grandezza
      do
        {
         //--- apre il file
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
```

```
            {
             //--- riceve la grandezza del file
             file_size=FileSize(file_handle);
             //--- chiude il file
             FileClose(file_handle);
            }
          else
            {
             PrintFormat("Fallimento nell'aprire il file %s, Codice Errore = %d",file_n
             continue;
            }
          //--- fa il print della grandezza del file
          PrintFormat("La grandezza del file %s è uguale a %d byte",file_name,file_size
          //--- definisce il percorso per spostare il file
          string path;
          if(file_size>InpThresholdSize*1024)
             path=InpBigFolderName+"//"+file_name;
          else
             path=InpSmallFolderName+"//"+file_name;
          //--- sposta il file
          ResetLastError();
          if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
             PrintFormat("il file %s è stato spostato",file_name);
          else
             PrintFormat("Error, code = %d",GetLastError());
         }
       while(FileFindNext(search_handle,file_name));
       //--- chiude l'handle di ricerca
       FileFindClose(search_handle);
      }
    else
       Print("File non trovato!");
  }
```

FileTell

Il file restituisce la posizione corrente del puntatore di un file aperto.

```
ulong  FileTell(
   int  file_handle    // File handle
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

Valore restituito

Posizione attuale del descrittore del file in byte dall'inizio del file.

Nota

Per ottenere informazioni sull' errore chiamare GetLastError().

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                        Demo_FileTell.mq5 |
//|                                  Copyright 2013, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando lancia lo script
#property script_show_inputs
//--- parametri di input
input string InpFileName="file.txt";     // nome del file
input string InpDirectoryName="Data";    // nome della directory
input int    InpEncodingType=FILE_ANSI; // ANSI=32 or UNICODE=64
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- specifica il valore della variabile per generare numeri casuali
   _RandomSeed=GetTickCount();
//--- variabili per le posizioni dei punti di inizio della stringa
   ulong pos[];
   int   size;
//--- resetta il valore dell' errore
   ResetLastError();
//--- apre il file
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
```

```
     {
      PrintFormat("%s il file è disponibile per la lettura",InpFileName);
      //--- riceve la posizione d'inizio per ogni stringa nel file
      GetStringPositions(file_handle,pos);
      //--- definisce il numero di stringhe nel file
      size=ArraySize(pos);
      if(!size)
        {
         //--- si ferma se il file non ha stringhe
         PrintFormat("il file %s è vuoto!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- fa una selezione casuale di un numero di stringa
      int ind=MathRand()%size;
      //--- slitta la posizione al punto iniziale della stringa
      FileSeek(file_handle,pos[ind],SEEK_SET);
      //--- legge e fa il print della stringa con il numero ind
      PrintFormat("Testo della stringa con il numero %d: \"%s\"",ind,FileReadString(fi
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("il file %s è stato chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
//+------------------------------------------------------------------------------------+
//| La funzione definisce i punti di inizio per ogni stringa nel file e                |
//| li piazza nell'array arr                                                           |
//+------------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- grandezza array di default
   int def_size=127;
//--- alloca la memoria per l'array
   ArrayResize(arr,def_size);
//--- contatore stringa
   int i=0;
//--- se questa non è la fine del file, allora c'è ancora una stringa
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // il file è vuoto, esce
//--- definisce lo slittamento in byte a seconda della codifica
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
```

```
      shift=1;
   else
      shift=2;
//--- va attraverso le stringhe nel loop
   while(1)
     {
      //--- legge la stringa
      FileReadString(handle);
      //--- controlla la fine del file
      if(!FileIsEnding(handle))
        {
         //--- memorizza la posizione della prossima stringa
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- incrementa la grandezza dell'array se è stato riempito
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // fine del file, uscita
     }
//--- definisce l'attuale grandezza dell'array
   ArrayResize(arr,i);
  }
```

FileWrite

La funzione è intesa per la scrittura di dati in un file CSV, il delimitatore viene inserito automaticamente a meno che non è uguale a 0. Dopo aver scritto nel file, verrà aggiunto il carattere di fine riga "\r\n".

```
uint  FileWrite(
   int  file_handle,    // File handle
   ...                  // Elenco dei parametri registrati
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

...

[in] L'elenco dei parametri separati da virgole, per scrivere nel file. Il numero di parametri scritti può essere fino a 63.

Valore restituito

Numero di byte scritti.

Nota

I numeri saranno convertiti in un testo in uscita (vedere la funzione Print()). I dati di tipo double vengono emessi con la precisione di 16 cifre dopo la virgola, ed i dati possono essere visualizzati sia in formato tradizionale che in formato scientifico - in base a quale dei due formati, l'output sarà più compatto. I dati di tipo float sono mostrati con 5 cifre dopo la virgola decimale. Per emettere in output i numeri reali con precisione diversa o in un formato chiaramente specificato, utilizzare DoubleToString().

I numeri di tipo bool vengono visualizzati come le stringhe "true" o "false". I numeri del tipo datetime vengono visualizzati come "AAAA.MM.GG. HH:MI:SS".

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                        Demo_FileWrite.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri per la ricezione dei dati dal terminale
input string             InpSymbolName="EURUSD";          // coppia di valute
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;       // time frame
input int                InpFastEMAPeriod=12;             // periodo EMA veloce
input int                InpSlowEMAPeriod=26;             // periodo EMA lento
```

```
input int                InpSignalPeriod=9;                // differenza tra periodi d
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo di prezzo
input datetime           InpDateStart=D'2012.01.01 00:00'; // data di inzio copiatura
//--- parametri per scrivere i dati nel file
input string             InpFileName="MACD.csv";  // nome file
input string             InpDirectoryName="Data"; // nome directory
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   datetime date_finish; // data di fine copiatura dati
   bool     sign_buff[]; // array signal (true - buy, false - sell)
   datetime time_buff[]; // array dell'orario di arrivo di signal
   int      sign_size=0; // grandezza array signal
   double   macd_buff[]; // array dei valori indicatore
   datetime date_buff[]; // array date indicatore
   int      macd_size=0; // grandezza array indicatore
//--- orario di fine è l'ora corrente
   date_finish=TimeCurrent();
//--- riceve l'handle dell'indicatore MACD
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- fallimento nel ricevere l'handle indicatore
      PrintFormat("Errore quando si è ricevuto l'handle indicatore. Codice Errore = %d
      return;
     }
//--- essendo nel ciclo finché l'indicatore calcola tutti i valori
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // pausa per consentire all'indicatore di calcolare tutti i suoi valo
//--- copia i valori dell'indicatore per un certo periodo di tempo
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori indicatore. Codice Errore = %d",Get
      return;
     }
//--- copia l'orario appropriato per i valori indicatore
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori temporali. Codice Errore = %d",GetI
      return;
     }
//--- libera la memoria occupata per l'indicatore
   IndicatorRelease(macd_handle);
//--- riceve la grandezza del buffer
```

```
   macd_size=ArraySize(macd_buff);
//--- analizza i dati e salva i segnali indicatore nell'array
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- segnale buy
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- segnale sell
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- apre il file per la scrittura dei valori indicatore (se il file è assente, verrà
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s è disponibile per la scrittura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- prima, scrive il numero di segnali
      FileWrite(file_handle,sign_size);
      //--- scrive l'orario ed i valori dei segnali al file
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

Commento, Stampa, StringFormat

FileWriteArray

La funzione scrive array di qualsiasi tipo ad eccezione di stringa ad un file BIN (può essere un array di strutture che non contengono stringhe o array dinamici).

```
uint  FileWriteArray(
   int           file_handle,          // File handle
   const void&   array[],              // Array
   int           start=0,              // Indice start nell'array
   int           count=WHOLE_ARRAY     // Numero di elementi
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

array[]

[out] Array per la registrazione.

start=0

[in] Indice iniziale nell' array (numero del primo elemento registrato).

count=WHOLE_ARRAY

[in] Numero di elementi da scrivere (WHOLE_ARRAY significa che tutti gli elementi iniziano con il numero start fino alla fine dell'array, verranno scritti).

Valore restituito

Numero di elementi registrati.

Nota

Array di stringhe possono essere registrati in un file TXT. In questo caso, le stringhe vengono automaticamente chiuse dai caratteri di fine riga "\r\n". A seconda del tipo di file ANSI o UNICODE, le stringhe sono o convertite in codifica-ansi o meno.

Esempio:

```
//+------------------------------------------------------------------------------------+
//|                                                      Demo_FileWriteArray.mq5 |
//|                                  Copyright 2013, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parametri di input
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------------------------+
//| Struttura per memorizzare i dati dei prezzi |
//+------------------------------------------------------------------------------------+
```

```
struct prices
  {
   datetime          date; // date
   double            bid;  // prezzo bid
   double            ask;  // prezzo ask
  };
//--- variabili globali
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- alloca la memoria per l'array
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                                           |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- scrive le stringhe di conteggio rimanenti se count<n
   WriteData(count);
  }
//+------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                    |
//+------------------------------------------------------------------------------+
void OnTick()
  {
//--- salva i dati nell'array
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- mostra i correnti dati
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
//--- incrementa il contatore
   count++;
//--- se l'array è riempito, scrive i dati nel file e lo azzera
   if(count==size)
     {
      WriteData(size);
      count=0;
     }
  }
```

```
//+------------------------------------------------------------------------------+
//| Scrive n elementi dell'array nel file                                        |
//+------------------------------------------------------------------------------+
void WriteData(const int n)
  {
//--- apre il file
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
   if(handle!=INVALID_HANDLE)
     {
      //--- scrive i dati array alla fine del file
      FileSeek(handle,0,SEEK_END);
      FileWriteArray(handle,arr,0,n);
      //--- chiude il file
      FileClose(handle);
     }
   else
      Print("Fallimento nell'aprire il file, errore",GetLastError());
  }
```

Vedi anche

Variabili, FileSeek

FileWriteDouble

La funzione scrive il valore di un parametro double ad un file-bin, a partire dalla posizione corrente del puntatore del file.

```
uint  FileWriteDouble(
   int     file_handle,     // File handle
   double  value            // Valore da scrivere
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

valore

[in] Il valore di tipo double.

Valore restituito

In caso di successo la funzione restituisce il numero di byte scritti (in questo caso sizeof(double)=8). Il puntatore del file viene spostato dello stesso numero di byte.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                  Demo_FileWriteDouble.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri per la ricezione dei dati dal terminale
input string             InpSymbolName="EURJPY";           // coppia di valute
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_M15;       // time frame
input int                InpMAPeriod=10;                   // periodo smoothing
input int                InpMAShift=0;                     // slittamento indicatore
input ENUM_MA_METHOD     InpMAMethod=MODE_SMA;             // tipo di smoothing
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo di prezzo
input datetime           InpDateStart=D'2013.01.01 00:00'; // data di inizio copiatura
//--- parameters for writing data to the file
input string             InpFileName="MA.csv";     // nome del file
input string             InpDirectoryName="Data"; // nome directory
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
```

```
   datetime date_finish=TimeCurrent();
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- riceve l'handle dell'indicatore MA
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- fallimento nel ricevere l'handle dell'indicatore
      PrintFormat("Errore quando si è ricevuto l'handle indicatore. Codice Errore = %d
      return;
     }
//--- essendo nel ciclo finché l'indicatore calcola tutti i valori
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // una pausa per consentire all'indicatore di calcolare tutti i suoi
   PrintFormat("I valori dell'indicatore che iniziano da %s verranno scritti nel file'
//--- copia i valori indicatore
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori dell'indicatore. Codice Errore = %d
      return;
     }
//--- copia l'orario di arrivo delle barre appropriate
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori temporali. Codice Errore = %d",GetI
      return;
     }
//--- riceve la grandezza del buffer
   size=ArraySize(ma_buff);
//--- libera la memoria occupata per l'indicatore
   IndicatorRelease(ma_handle);
//--- apre il file per la scrittura dei valori indicatore (se il file è assente, verrà
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s è disponibile per la scrittura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- prima, scrive la grandezza del campione dati
      FileWriteDouble(file_handle,(double)size);
      //--- scrive l'orario ed il valore dell'indicatore nel file
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
```

```
         }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

I tipi reali (double, float)

FileWriteFloat

La funzione scrive il valore di un parametro float ad un file-bin, a partire dalla posizione corrente del puntatore del file.

```
uint  FileWriteFloat(
   int    file_handle,     // File handle
   float  value            // Valore da scrivere
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

valore

[in] Il valore di tipo float.

Valore restituito

In caso di successo la funzione restituisce il numero di byte scritti (in questo caso sizeof(Float)=4). Il puntatore del file viene spostato dello stesso numero di byte.

Esempio:

```
//+------------------------------------------------------------------------------------+
//|                                                        Demo_FileWriteFloat.mq5 |
//|                                  Copyright 2013, MetaQuotes Software Corp. |
//|                                                             https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri per la ricezione dei dati dal terminale
input string          InpSymbolName="EURUSD";           // coppia di valute
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // time frame
input datetime        InpDateStart=D'2013.01.01 00:00'; // data di inzio copiatura dat
//--- parameters for writing data to the file
input string          InpFileName="Close.bin"; // nome file
input string          InpDirectoryName="Data"; // nome directory
//+------------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------------+
voidOnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
```

```
//--- resetta il valore dell' errore
   ResetLastError();
//--- copia i prezzi di chiusura per ogni barra
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Fallimento nel copiare i valori di prezzo di chiusura (close). Codi
      return;
     }
//--- copia l'orario per ogni barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori tempo. Codice Errore = %d",GetLastE
      return;
     }
//--- riceve la grandezza del buffer
   size=ArraySize(close_buff);
//--- apre il file per la scrittura dei valori (se il file è assente, verrà automatica
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIl
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s viene aperto per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- scrive orario e valore dei prezzi di chiusura close, nel file
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

I Tipi Reali (double, float), FileWriteDouble

FileWriteInteger

La funzione scrive il valore del parametro int ad un file-bin, a partire dalla posizione corrente del puntatore del file.

```
uint  FileWriteInteger(
   int  file_handle,         // File handle
   int  value,               // Valore da scrivere
   int  size=INT_VALUE       // Grandezza in byte
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

valore

[in] Valore integer.

size=INT_VALUE

[in] Numero di byte (fino a 4 inclusivi), che devono essere scritti. Le costanti sono disponibili le corrispondenti: CHAR_VALUE=1, SHORT_VALUE=2 ed INT_VALUE=4, per cui la funzione può scrivere il valore integer di tipo char, uchar, short, ushort, int o uint.

Valore restituito

In caso di successo la funzione restituisce il numero di byte scritti. Il puntatore del file viene spostato dello stesso numero di byte.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                  Demo_FileWriteInteger.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri per la ricezione dei dati dal terminale
input string              InpSymbolName="EURUSD";           // coppia di valute
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_H1;        // time frame
input datetime            InpDateStart=D'2013.01.01 00:00'; // data di inizio copiatura
//--- parameters for writing data to the file
input string              InpFileName="Trend.bin"; // nome file
input string              InpDirectoryName="Data"; // nome directory
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
```

```
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- resetta il valore dell' errore
   ResetLastError();
//--- copia i prezzi di chiusura per ogni barra
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Fallimento nel copiare i valori dei prezzi close. Codice Errore = %
      return;
     }
//--- copia l'orario per ogni barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori temporali. Codice Errore = %d",GetL
      return;
     }
//--- riceve la grandezza del buffer
   size=ArraySize(close_buff);
//--- apre il file per la scrittura dei valori (se il file è assente, verrà automatica
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s è disponibile per la scrittura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //---
      int   up_down=0; // trend flag
      int   arr_size;  // grandezza dell' array arr
      uchar arr[];     // array di tipo uchar
      //--- scrive i valori orari nel file
      for(int i=0;i<size-1;i++)
        {
         //--- confronta i prezzi di chiusura della barra corrente e successiva
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
               //--- scrive i valori data nel file usando FileWriteInteger
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- prima, scrive il numero di simboli nell'array
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- scrive i simboli
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- cambia il trend flag
```

```
                up_down=1;
               }
            }
         else
            {
             if(up_down!=-1)
               {
                //--- scrive i valori data nel file usando FileWriteInteger
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //--- prima, scrive il numero di simboli nell'array
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- scrive i simboli
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- cambia il trend flag
                up_down=-1;
               }
            }
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

IntegerToString, StringToInteger, Integer types

FileWriteLong

La funzione scrive il valore del parametro di tipo long ad un file-bin, a partire dalla posizione corrente del puntatore del file.

```
uint  FileWriteLong(
   int   file_handle,     // File handle
   long  value            // Valore da scrivere
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

valore

[in] Valore di tipo long.

Valore restituito

In caso di successo la funzione restituisce il numero di byte scritti (in questo caso sizeof(long)=8). Il puntatore del file viene spostato dello stesso numero di byte.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                    Demo_FileWriteLong.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri per la ricezione dei dati dal terminale
input string          InpSymbolName="EURUSD";           // coppia di valute
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // time frame
input datetime        InpDateStart=D'2013.01.01 00:00'; // data di inzio copiatura dat
//--- parameters for writing data to the file
input string          InpFileName="Volume.bin"; // nome file
input string          InpDirectoryName="Data";  // nome directory
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
```

```
//--- resetta il valore dell' errore
   ResetLastError();
//--- copia i volumi tick per ogni barra
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_bu
     {
      PrintFormat("Fallimento nel copiare i valori del volume tick. Codice Errore = %
      return;
     }
//--- copia l'orario per ogni barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori temporali. Codice Errore = %d",GetL
      return;
     }
//--- riceve la grandezza del buffer
   size=ArraySize(volume_buff);
//--- apre il file per la scrittura dei valori indicatore (se il file è assente, verra
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s è disponibile per la scrittura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- prima, scrive la grandezza dei dati campione
      FileWriteLong(file_handle,(long)size);
      //--- scrive i valori orari e di volume, nel file
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

Tipi Interi, FileWriteInteger

FileWriteString

La funzione scrive il valore di un parametro di tipo-stringa in un file BIN, CSV o TXT a partire dalla posizione corrente del puntatore del file. Quando si scrive in un file CSV o TXT: se c'è un simbolo nella stringa '\n' (LF) senza carattere precedente '\r' (CR), allora prima di '\n' viene agginto il '\r' mancante.

```
uint  FileWriteString(
   int           file_handle,     // File handle
   const string  text_string,     // Stringa da scrivere
   int           length=-1        // Numero di simboli
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

text_string

[in] String.

length=-1

[in] Il numero di caratteri che si desidera scrivere. Questa opzione è necessaria per scrivere una stringa in un file BIN. Se la dimensione non è specificata, allora viene scritta l'intera stringa senza il trailer 0. Se si specifica una dimensione inferiore alla lunghezza della stringa, allora viene scritta una parte della stringa senza il trailer 0. Se si specifica una dimensione maggiore della lunghezza della stringa, la stringa è riempita con il numero appropriato di zeri. Per i file di tipo CSV e TXT, questo parametro viene ignorato e la stringa viene scritta interamente.

Valore restituito

In caso di successo la funzione restituisce il numero di byte scritti. Il puntatore del file viene spostato dello stesso numero di byte.

Nota

Si noti che quando si scrive in un file aperto dal flag FILE_UNICODE (o senza un flag FILE_ANSI), allora il numero di byte scritti sarà due volte più grande del numero di caratteri stringa scritti. Quando si registra su un file aperto con il flag FILE_ANSI, il numero di byte scritti coinciderà con il numero di caratteri della stringa scritta.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                  Demo_FileWriteString.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri per la ricezione dei dati dal terminale
```

```
input string             InpSymbolName="EURUSD";           // coppia di valute
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // time frame
input int                InpMAPeriod=14;                   // periodo MA
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo di prezzo
input datetime           InpDateStart=D'2013.01.01 00:00'; // data di inizio copiatura
//--- parameters for writing data to the file
input string             InpFileName="RSI.csv";    // nome file
input string             InpDirectoryName="Data"; // nome directory
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   datetime date_finish; // data di fine copiatura dati
   double   rsi_buff[];  // array dei valori indicatore
   datetime date_buff[]; // array delle date indicatore
   int      rsi_size=0;  // grandezza degli array indicatore
//--- l'orario di fine è quello corrente
   date_finish=TimeCurrent();
//--- riceve l'handle dell'indicatore RSI
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- fallimento nel ricevere l'handle dell'indicatore
      PrintFormat("Errore quando si è ricevuto l'handle indicatore. Codice Errore = %d
      return;
     }
//--- è in loop, finchè l'indicatore calcola tutti i suoi valori
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // una pausa per consentire all'indicatore di calcolare tutti i suoi
//--- copia i valori dell'indicatore per un certo periodo di tempo
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori indicatore. Codice Errore = %d",Get
      return;
     }
//--- copia l'orario appropriato per i valori indicatore
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori temporali. Codice Errore = %d",GetI
      return;
     }
//--- libera la memoria occupata per l'indicatore
   IndicatorRelease(rsi_handle);
//--- riceve la grandezza del buffer
   rsi_size=ArraySize(rsi_buff);
```

```
//--- apre il file per la scrittura dei valori indicatore (se il file è assente, verrà
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s è disponibile per la scrittura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH))
      //--- prepara le variabili aggiuntive
      string str="";
      bool   is_formed=false;
      //--- scrive i dati per formare aree overbought ed oversold
      for(int i=0;i<rsi_size;i++)
        {
         //--- controlla i valori indicatore
         if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
           {
            //--- se il valore è il primo in quest'area
            if(!is_formed)
              {
               //--- aggiunge il valore e la data
               str=(string)rsi_buff[i]+"\t"+(string)date_buff[i];
               is_formed=true;
              }
            else
               str+="\t"+(string)rsi_buff[i]+"\t"+(string)date_buff[i];
            //--- sposta alla prossima iterazione del loop
            continue;
           }
         //--- controlla il flag
         if(is_formed)
           {
            //--- la stringa è formata, viene scritta nel file
            FileWriteString(file_handle,str+"\r\n");
            is_formed=false;
           }
        }
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

Tipo Stringa, StringFormat

FileWriteStruct

La funzione scrive nel contenuto di un file-bin una struttura passata come parametro, partendo dalla posizione corrente del puntatore del file.

```
uint  FileWriteStruct(
   int           file_handle,       // File handle
   const void&   struct_object,     // Link ad un oggetto
   int           size=-1            // Grandezza da scrivere in byte
   );
```

Parametri

file_handle

[in] Il descrittore di file restituito da FileOpen().

struct_object

[in] Riferimento all'oggetto di questa struttura. La struttura non deve contenere stringhe, array dinamici o funzioni virtuali.

size=-1

[in] Numero di byte che si desidera registrare. Se la grandezza non è specificata o il numero specificato di byte è maggiore della grandezza della struttura, viene scritta l'intera struttura.

Valore restituito

In caso di successo la funzione restituisce il numero di byte scritti. Il puntatore del file viene spostato dello stesso numero di byte.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                    Demo_FileWiteStruct.mq5 |
//|                                Copyright 2013, MetaQuotes Software Corp. |
//|                                                    https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostra la finestra dei parametri di input quando si lancia lo script
#property script_show_inputs
//--- parametri per la ricezione dei dati dal terminale
input string          InpSymbolName="EURUSD";           // coppia di valute
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // time frame
input datetime        InpDateStart=D'2013.01.01 00:00'; // data di inzio copiatura dat
//--- parameters for writing data to the file
input string          InpFileName="EURUSD.txt";         // nome del file
input string          InpDirectoryName="Data";          // nome directory
//+------------------------------------------------------------------------------+
//| Struttura per memorizzare i dati della candela                               |
//+------------------------------------------------------------------------------+
struct candlesticks
```

```
  {
   double            open;  // prezzo apertura
   double            close; // prezzo chiusura
   double            high;  // prezzo più alto
   double            low;   // prezzo più basso
   datetime          date;  // data
  };
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- resetta il valore dell' errore
   ResetLastError();
//--- riceve l'orario di arrivo delle barre dal range
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori temporali. Codice Errore = %d",GetL
      return;
     }
//--- riceve i prezzi high delle barre, dal range
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori dei prezzi high. Codice Errore = %d
      return;
     }
//--- riceve i valori low delle barre dal range
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori dei prezzi low. Codice Errore = %d"
      return;
     }
//--- riceve i prezzi open delle barre, dal range
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("Fallimento nel copiare i valori dei prezzi open. Codice Errore = %d
      return;
     }
//--- riceve i prezzi close delle barre dal range
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
```

```
      PrintFormat("Fallimento nel copiare i valori dei prezzi close. Codice Errore = %
      return;
     }
//--- definisce la dimensione degli array
   size=ArraySize(time_buff);
//--- salva i dati nella array struttura
   ArrayResize(cand_buff,size);
   for(int i=0;i<size;i++)
     {
      cand_buff[i].open=open_buff[i];
      cand_buff[i].close=close_buff[i];
      cand_buff[i].high=high_buff[i];
      cand_buff[i].low=low_buff[i];
      cand_buff[i].date=time_buff[i];
     }

//--- apre il file per la scrittura della struttura array nel file (se il file è asser
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("il file %s viene aperto per la lettura",InpFileName);
      PrintFormat("Percorso file: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_
      //--- prepara il contatore del numero di byte
      uint counter=0;
      //--- scrive i valori array nel loop
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat("%d byte di informazione vengono scritti nel file %s",InpFileName,co
      PrintFormat("Numero totale di byte: %d * %d * %d = %d, %s",size,5,8,size*5*8,siz
      //--- chiude il file
      FileClose(file_handle);
      PrintFormat("I dati vengono scritti, il file %s è chiuso",InpFileName);
     }
   else
      PrintFormat("Fallimento nell'aprire il file %s, Codice errore = %d",InpFileName,
  }
```

Vedi anche

Le strutture e le classi

FileLoad

Legge tutti i dati di un file binario specificato in un array passato, di tipo numerico o strutture semplici. La funzione consente di leggere rapidamente i dati di tipo noto nell' array appropriato.

```
long  FileLoad(
   const string  file_name,          // Nome del File
   void&         buffer[],           // Un array di tipo numerico o strutture semplici
   int           common_flag=0       // Un flag file, viene cercato in <data_folder>\MQ
   );
```

Parametri

file_name

[in] Il nome del file da cui vengono letti i dati.

buffer

[in] Un array di tipo numerico o strutture semplici.

common_flag=0

[in] Una flag file indicante la modalità di funzionamento. Se il parametro non viene specificato, il file viene cercato nella sottocartella MQL5\Files (o in <testing_agent_directory>\MQL5\Files in caso di testing).

Valore di ritorno

Il numero di elementi letti oppure -1 in caso di errore.

Nota

La funzione FileLoad() legge da un file il numero di byte multiplo della grandezza dell'array. Supponiamo che la dimensione del file è di 10 byte, e la funzione legge i dati in un array di tipo double (sizeof(Double)=8). In questo caso la funzione leggerà solo 8 byte, i restanti 2 byte alla fine del file verranno scartati, e la funzione FileLoad() restituirà 1 (1 elemento letto).

Esempio:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileLoad.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- parametri di input
input int      bars_to_save=10; // Numero di barre
//+------------------------------------------------------------------+
```

```
//| Script program funzione start                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_rates.bin";
   MqlRates rates[];
//---
   int copied=CopyRates(_Symbol,_Period,0,bars_to_save,rates);
   if(copied!=-1)
     {
      PrintFormat(" CopyRates(%s) copied %d bars",_Symbol,copied);
      //---  Scrive quotazioni sul file
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyRates(%s), error=",_Symbol,GetLastError());
// --- Ora legge queste quotazioni indietro al file
   ArrayFree(rates);
   long count=FileLoad(filename,rates,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tOpen\tHigh\tLow\tClose\tTick Voulme\tSpread\tReal Volume");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s\t%G\t%G\t%G\t%G\t%I64u\t%d\t%I64u",
                     TimeToString(rates[i].time,TIME_DATE|TIME_SECONDS),
                     rates[i].open,rates[i].high,rates[i].low,rates[i].close,
                     rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
  }
```

Guarda anche

Strutture e Classi, FileReadArray, FileReadStruct, FileSave

FileSave

Scrive in un file binario tutti gli elementi di un array passato come parametro. La funzione consente di scrivere rapidamente array di tipo numerico o strutture semplici come una stringa.

```
bool  FileSave(
   const string  file_name,          // Nome del File
   void&         buffer[],           // Un array di tipo numerico o strutture semplici
   int           common_flag=0       // Un flag file, per default i files vengono scrit
   );
```

Parametri

file_name

[in] Il nome del file, in cui i dati dell'array verranno scritti.

buffer

[in] Un array di tipo numerico o strutture semplici.

common_flag=0

[in] Una flag file indicante la modalità di funzionamento. Se il parametro non viene specificato, il file verrà scritto nella sottocartella MQL5\Files (o per <testing_agent_directory>\MQL5\Files in caso di testing).

Valore di ritorno

In caso di fallimento restituisce false.

Esempio:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSave.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- parametri di input
input int      ticks_to_save=1000; // Numero di ticks
//+------------------------------------------------------------------+
//| Script program funzione start                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_ticks.bin";
   MqlTick ticks[];
//---
   int copied=CopyTicks(_Symbol,ticks,COPY_TICKS_ALL,0,ticks_to_save);
   if(copied!=-1)
     {
```

```
      PrintFormat(" CopyTicks(%s) copied %d ticks",_Symbol,copied);
      //--- Se lo storico dei tick è sincronizzato, il codice errore code è uguale a z
      if(!GetLastError()==0)
         PrintFormat("%s: I Ticks non sono sincronizzati, error=%d",_Symbol,copied,_La
      //---  Scrive i ticks su un file
      if(!FileSave(filename,ticks,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyTicks(%s), Error=",_Symbol,GetLastError());
//--- Ora legge i ticks indietro al file
   ArrayFree(ticks);
   long count=FileLoad(filename,ticks,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tBid\tAsk\tLast\tVolume\tms\tflags");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s.%03I64u:\t%G\t%G\t%G\t%I64u\t0x%04x",
         TimeToString(ticks[i].time,TIME_DATE|TIME_SECONDS),ticks[i].time_msc%1000,
         ticks[i].bid,ticks[i].ask,ticks[i].last,ticks[i].volume,ticks[i].flags);
        }
     }
  }
```

Guarda anche

Strutture e Classi, FileWriteArray, FileWriteStruct, FileLoad, FileWrite

FolderCreate

La funzione crea una cartella nella directory Files (a seconda del valore di common_flag).

```
bool  FolderCreate(
   string  folder_name,       // La stringa con il nome della nuova cartella
   int     common_flag=0      // Ambito
   );
```

Parametri

folder_name

[in] Il nome della directory che si desidera creare. Contains the relative path to the folder.

common_flag=0

[in]Flag determina la posizione della directory. Se common_flag=FILE_COMMON, la directory si trova nella cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, la directory si trova in una cartella locale (MQL5\files o MQL5\tester\file in caso di testing).

Valore restituito

Restituisce true se ha successo, altrimenti - false.

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Esempio:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- description
#property description "The script shows FolderCreate() application sample."
#property description "The external parameter defines the directory for creating folde
#property description "The folder structure is created after executing the script"

//--- display window of the input parameters during the script's launch
#property script_show_inputs
//--- the input parameter defines the folder, in which the script works
input bool     common_folder=false; // common folder for all terminals
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- folder to be created in MQL5\Files
   string root_folder="Folder_A";
   if(CreateFolder(root_folder,common_folder))
     {
```

```
      //--- create the Child_Folder_B1 sub-folder in it
      string folder_B1="Child_Folder_B1";
      string path=root_folder+"\\"+folder_B1;          // create the folder name consi
      if(CreateFolder(path,common_folder))
        {
         //--- create 3 more sub-directories in this folder
         string folder_C11="Child_Folder_C11";
         string child_path=root_folder+"\\"+folder_C11;// create the folder name consi
         CreateFolder(child_path,common_folder);
         //--- second sub-directory
         string folder_C12="Child_Folder_C12";
         child_path=root_folder+"\\"+folder_C12;
         CreateFolder(child_path,common_folder);

         //--- third sub-directory
         string folder_C13="Child_Folder_C13";
         child_path=root_folder+"\\"+folder_C13;
         CreateFolder(child_path,common_folder);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Try creating a folder and display a message about that           |
//+------------------------------------------------------------------+
bool CreateFolder(string folder_path,bool common_flag)
  {
   int flag=common_flag?FILE_COMMON:0;
   string working_folder;
//--- define the full path depending on the common_flag parameter
   if(common_flag)
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
   else
      working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- debugging message
   PrintFormat("folder_path=%s",folder_path);
//--- attempt to create a folder relative to the MQL5\Files path
   if(FolderCreate(folder_path,flag))
     {
      //--- display the full path for the created folder
      PrintFormat("Created the folder %s",working_folder+"\\"+folder_path);
      //--- reset the error code
      ResetLastError();
      //--- successful execution
      return true;
     }
   else
      PrintFormat("Failed to create the folder %s. Error code %d",working_folder+folde
//--- execution failed
```

```
   return false;
  }
```

Vedi anche

FileOpen(), FolderClean(), FileCopy()

FolderDelete

La funzione rimuove la directory specificata. Se la cartella non è vuota, allora non può essere rimossa.

```
bool  FolderDelete(
   string  folder_name,       // Stringa con il nome della cartella da eliminare
   int     common_flag=0      // Ambito
   );
```

Parametri

folder_name

[in] Il nome della directory che si desidera eliminare. Contiene il percorso completo della cartella.

common_flag=0

[in]Flag determina la posizione della directory. Se common_flag=FILE_COMMON, la directory si trova nella cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, la directory si trova in una cartella locale (MQL5\file o MQL5\tester\file in caso di testing).

Valore restituito

Restituisce true se ha successo, altrimenti false.

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Se la directory contiene almeno un file e/o sottodirectory, allora questa directory non può essere eliminata, deve essere prima pulita. FolderClean() è usata per pulire una cartella di tutti i suoi file o sottocartelle.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                      Demo_FolderDelete.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Descrizione
#property description "Lo script mostra un esempio di uso di FolderDelete()."
#property description "Le prime due cartelle sono state create; una di esse è vuota, l
#property description "Quando si prova ed eliminare la cartella non vuota, un errore v

//--- Mostra la finestra dei parametri di input quando inizia lo script
#property script_show_inputs
//--- parametri di Input
input string   firstFolder="empty";     // La cartella vuota
input string   secondFolder="nonempty";// La cartella, nel quale un file è stato creat
```

```
string filename="delete_me.txt";       // Il nome del file che verrà creato nella cart
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- Scrive l'handle del file qui
   int handle;
//--- Rileva in quale cartella stiamo lavorando
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- Il messaggio di debug
   PrintFormat("working_folder=%s",working_folder);
//--- Prova a creare una cartella vuota relativa al percorso MQL5\Files
   if(FolderCreate(firstFolder,0)) // 0 significa che stiamo lavorando nella cartella
     {
      //--- Introduce il percorso completo per la cartella creata
      PrintFormat("La cartella %s è stata creata",working_folder+"\\"+firstFolder);
      //--- Resetta il codice errore
      ResetLastError();
     }
   else
      PrintFormat("Fallimento nel creare la cartella %s. Codice errore %d",working_fol

//--- Ora crea una cartella non vuota usando la funzioneFileOpen()
   string filepath=secondFolder+"\\"+filename;  // Percorso al file che vogliamo apri
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // Flag FILE_WRITE in questo vaso è
   if(handle!=INVALID_HANDLE)
      PrintFormat("Il file %s è stato aperto per la lettura",working_folder+"\\"+filep
   else
      PrintFormat("Fallimento nel creare il file %s nella cartella %s. Error code=",fi

   Comment(StringFormat("Prepare to delete folders %s and %s", firstFolder, secondFold
//--- Una piccola pausa di 5 secondi per leggere il messaggio nel chart
   Sleep(5000); // Sleep() non può essere usato negli indicatori!

//--- Mostra una finestra e chiede all'utente
   int choice=MessageBox(StringFormat("Vuoi eliminare le cartelle %s ed %s?", firstFol
                         "Eliminazione cartelle",
                         MB_YESNO|MB_ICONQUESTION); //  Due bottoni - "Yes" e "No"

//--- Esegue un'azione a seconda della variante scelta
   if(choice==IDYES)
     {
      //--- Elimina il commento dal chart
      Comment("");
      //--- Aggiunge il messaggio nel journal "Experts"
      PrintFormat("Prova ad eliminare le cartelle %s ed %s",firstFolder, secondFolder)
      ResetLastError();
      //--- Elimina le cartelle vuote
      if(FolderDelete(firstFolder))
```

```
         //--- Il seguente messaggio dovrebbe apparire giacchè la cartella è vuota
         PrintFormat("La cartella %s è stata eliminata con successo",firstFolder);
      else
         PrintFormat("Fallimento nell'eliminare la cartella %s. Codice errore=%d", fi
      ResetLastError();
      //--- Elimina la cartella che contiene il file
      if(FolderDelete(secondFolder))
         PrintFormat("La cartella %s è stata eliminata con successo", secondFolder);
      else
         //--- Il seguente messaggio dovrebbe apparire giacchè la cartella contiene u
         PrintFormat("Fallimento nell'eliminare la cartella %s. Codice Errore=%d", se
     }
   else
      Print("Eliminazione annullata");
//---
  }
```

Vedi anche

FileOpen(), FolderClean(), FileMove()

FolderClean

La funzione cancella tutti i file in una cartella specificata.

```
bool  FolderClean(
   string  folder_name,       // La stringa con il nome della cartella eliminata
   int     common_flag=0      // Ambito
   );
```

Parametri

folder_name

[in] Il nome della directory in cui si desidera eliminare tutti i file. Contiene il percorso completo della cartella.

common_flag=0

[in]Flag determina la posizione della directory. Se common_flag = FILE_COMMON, la directory si trova nella cartella condivisa per tutti i terminali client \Terminal\Common\Files. In caso contrario, la directory si trova in una cartella locale (MQL5\files o MQL5\tester\files in caso di testing).

Valore restituito

Restituisce true se ha successo, altrimenti false.

Nota

Per motivi di sicurezza, il lavoro con i file è strettamente controllato nel linguaggio MQL5. I file con cui sono condotte le operazioni di file utilizzando i mezzi MQL5, non possono esservi al di fuori della sandbox del file.

Questa funzione deve essere usata con cautela, dal momento che tutti i file e tutte le sottodirectory vengono eliminate irrimediabilmente.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                        Demo_FolderClean.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Descrizione
#property description "Lo script mostra un esempio di uso di FolderClean()."
#property description "Prima, i file vengono creati nella cartella specificata usando
#property description "Quindi, prima che i file vengano eliminati, un avviso viene mos

//--- Mostra la finestra di dialogo dei parametri di input quando inizia lo script
#property script_show_inputs
//--- parametri di Input
input string   foldername="demo_folder";  // Crea una cartella in MQL5/Files/
input int      files=5;                   // Il numero di file per creare ed eliminare
```

```
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
   string name="testfile";
//--- Prima apre o crea i file nella cartella dati del terminale
   for(int N=0;N<files;N++)
     {
      //--- Il nome del file in forma di 'demo_folder\testfileN.txt'
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- Apre il file con il flag per la scrittura, in questo caso, la cartella 'de
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- Rileva se la funzione FileOpen() ha avuto successo
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("Fallimento nel creare il file %s. Error code",filemane,GetLastEr
         ResetLastError();
        }
      else
        {
         PrintFormat("Il file %s è stato aperto con successo",filemane);
         //--- Il file aperto non serve più, quindi viene chiuso
         FileClose(handle);
        }
     }

//--- Controlla il numero di file nella cartella
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("La cartella %s contiene in totale %d file",foldername,k);

//--- Mostra una finestra per chiedere all'utente
   int choice=MessageBox(StringFormat("Stai per eliminare %d file dalla cartella %s. S
                         "Elimina i file dalla cartella",
                         MB_YESNO|MB_ICONQUESTION); //  Due bottoni - "Si" e "No" ("Ye
   ResetLastError();

//--- Esegue l'azione a seconda della variante selezionata
   if(choice==IDYES)
     {
      //--- Inizia ad eliminare i file
      PrintFormat("Prova ad eliminare tutti i file dalla cartella %s",foldername);
      if(FolderClean(foldername,0))
         PrintFormat("I file sono stati eliminati con successo, %d file rimasti nella
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("Fallimento nell'eliminare i file dalla cartella %s. Codice erro
     }
   else
```

```
      PrintFormat("Eliminazione annullata");
//---
  }
//+------------------------------------------------------------------------------------+
//| Restituisce il numero di file nella cartella specificata                           |
//+------------------------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- Se almeno un file viene trovato, cerca per più file
   if(handle!=INVALID_HANDLE)
     {
      //--- Mostra il nome del file
      PrintFormat("Il file %s è stato trovato",filename);
      //--- Incrementa il contatore dei file/cartelle trovati
      count++;
      //--- Inizia la ricerca in tutti i file/cartelle
      while(FileFindNext(handle,filename))
        {
         PrintFormat("Il file %s è stato trovato",filename);
         count++;
        }
      //--- Non dimenticare di chiudere l'handle di ricerca una volta completata
      FileFindClose(handle);
     }
   else // Fallimento nell'ottenere l'handle
     {
      PrintFormat("Ricerca file nella cartella %s fallita",path);
     }
//--- Restituisce il risultato
   return count;
  }
```

Vedi anche

FileFindFirst, FileFindNext, FileFindClose

Custom Indicators

Questo è il gruppo funziona utilizzati nella creazione di indicatori personalizzati. Queste funzioni non possono essere utilizzate durante la scrittura di Expert Advisors e Script.

Funzione	Azione
SetIndexBuffer	Associa il buffer indicatore specificato con l' array dinamico uni-dimensionale di tipo double
IndicatorSetDouble	Imposta il valore della proprietà dell' indicatore di tipo double
IndicatorSetInteger	Imposta il valore di una proprietà dell'indicatore di tipo int
IndicatorSetString	Imposta il valore della proprietà dell' indicatore di tipo string
PlotIndexSetDouble	Imposta il valore della proprietà dell' indicatore di tipo double
PlotIndexSetInteger	Imposta il valore di una proprietà dell'indicatore di tipo int
PlotIndexSetString	Imposta il valore di una proprietà dell'indicatore di tipo string
PlotIndexGetInteger	Restituisce il valore della proprietà della linea dell'indicatore di tipo integer

La proprietà dell'indicatore può essere impostata utilizzando le direttive del compilatore o utilizzando le funzioni. Per capire meglio ciò, si consiglia di studiare gli stili indicatore negli esempi.

Tutti i calcoli necessari di un indicatore personalizzato devono essere messi nella funzione predeterminata OnCalculate(). Se si utilizza una forma breve della chiamata di funzione OnCalculate(), come

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

allora la variabile *rates_total* contiene il valore del numero totale di elementi dell'array prezzo[] passato come parametro di input per il calcolo dei valori degli indicatori.

Il parametro *prev_calculated* è il risultato dell'esecuzione di OnCalculate() alla chiamata precedente, ma permette di organizzare un algoritmo risparmiante risorse, per calcolare i valori degli indicatori. Ad esempio, se il valore corrente rates_total = 1000, prev_calculated = 999, allora forse è sufficiente per fare calcoli solo per un valore di ciascun buffer indicatore.

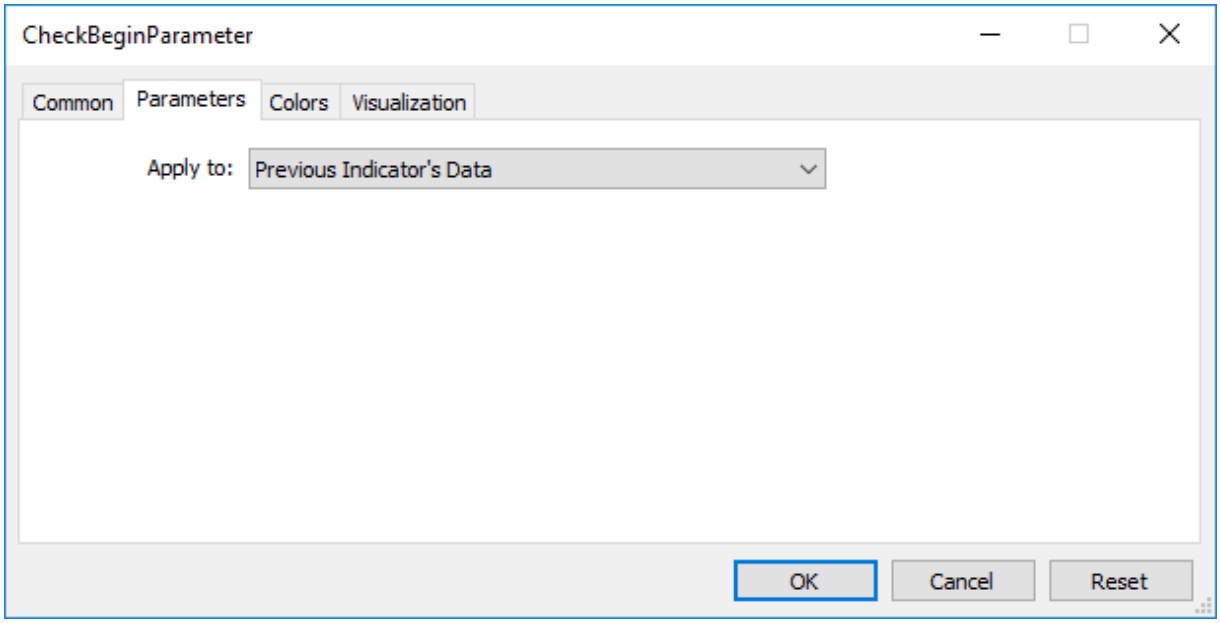
Se le informazioni rigardo la grandezza dell'array di input prezzi sarebbe stato non-disponibile, allora avrebbe portato alla necessità di fare calcoli per 1000 valori di ogni buffer indicatore. Alla prima chiamata di OnCalculate() Valore prev_calculated = 0. Se l'array prezzo[] è cambiato in qualche modo, allora in questo caso prev_calculated è anche uguale a 0.

Il parametro *begin* mostra il numero di valori iniziali dell'array prezzo, che non contengono dati per il calcolo. Per esempio, se i valori di Oscillator Accelerator (per cui i primi 37 valori non vengono calcolati) sono stati utilizzati come parametro di input, allora begin = 37. Ad esempio, consideriamo un semplice indicatore:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
```

```
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers indicatore
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato  |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

  {
//---
   Print("begin = ",begin,"  prev_calculated = ",prev_calculated,"  rates_total = ",ra
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Trascinandolo dalla finestra "Navigatore" alla finestra dell'indicatore Accelerator Oscillator, indicheremo che i calcoli saranno effettuati in base ai valori dell'indicatore precedente:



Come risultato, la prima chiamata di OnCalculate() avrà il valore di prev_calculated uguale a zero, e con ulteriori chiamate sarà uguale al valore rates_total (finché il numero di barre sul grafico dei prezzi aumenti).

Il valore del parametro begin sarà esattamente uguale al numero di barre iniziali, per cui i valori dell'indicatore Accelerator non vengono calcolati secondo la logica di questo indicatore. Se guardiamo il codice sorgente del indicatore personalizzato Accelerator.mq5, vedremo le seguenti linee nella funzione OnInit():

```
//--- imposta la prima barra da quell' indice sarà disegnato
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

Utilizzando la funzione PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, Empty_first_values), abbiamo impostato il numero di primi-valori non esistenti nell'array indicatore zero di un indicatore personalizzato, che non abbiamo bisogno di accettare per il calcolo (empty_first_values). Così, abbiamo meccanismi per:

1. impostare il numero di valori iniziali di un indicatore, che non dovrebbe essere utilizzato per i calcoli in un altro indicatore personalizzato;
2. ottenere informazioni sul numero dei primi valori per essere ignorati quando si chiama un altro indicatore personalizzato, senza entrare nella logica dei suoi calcoli.

Stili Indicatore negli Esempi

Il Terminale Client di MetaTrader 5 comprende 38 indicatori tecnici che possono essere utilizzati in programmi con MQL5 attraverso apposite funzioni. Ma il vantaggio principale del linguaggio MQL5 è la capacità di creare indicatori personalizzati, che possono poi essere utilizzati in Expert Advisor o semplicemente applicati su grafici dei prezzi allo scopo di analisi tecnica.

L'intero set di indicatori può essere derivata da vari stili di disegno base, conosciuti come plotting. Il plotting denota un modo di visualizzazione i dati, che l'indicatore calcola, immagazzina e fornisce a richiesta. Vi sono sette tali tipi di plotting fondamentali:

1. Una linea
2. Una sezione (segmento)
3. Istogramma
4. Freccia (simbolo)
5. Una zona verniciata (canale riempito)
6. Bars
7. Candele giapponesi

Ogni plotting richiede da uno a cinque array di tipo double, in cui sono memorizzati i valori degli indicatori. Allo scopo di convenienza, questi array sono associati con i buffer indicatore. Il numero di buffer in un indicatore deve essere dichiarato in anticipo utilizzando le direttive del compilatore, per esempio:

```
#property indicator_buffers 3 // Numero di buffers
#property indicator_plots   2 // numero di plots
```

Il numero di buffer in dell'indicatore è sempre maggiore o uguale al numero di plots nell'indicatore.

Poiché ogni tipo plotting base può avere variazione di colore o specifiche di costuzione, il numero effettivo di tipi di plotting in MQL5 è 18:

Plotting	Descrizione	Valore buffer	Color buffers
DRAW_NONE	Non è visivamente mostrato nel grafico, ma i valori del buffer corrispondente possono essere visualizzati nella	1	-

Plotting	Descrizione	Valore buffer	Color buffers
	finestra dei dati		
DRAW_LINE	Una linea viene tracciata sui valori del buffer corrispondente (i valori vuoti nel buffer sono indesiderabili)	1	-
DRAW_SECTION	È disegnato come segmenti di linea tra i valori del buffer corrispondente (di solito ha svariati valori vuoti)	1	-
DRAW_HISTOGRAM	È rappresentato con un istogramma dalla linea zero ai valori del buffer corrispondente (può avere valori vuoti)	1	-

Plotting	Descrizione	Valore buffer	Color buffers
DRAW_HISTOGRAM2	È disegnato come un istogramma basato su due buffer indicatori (può avere valori vuoti)	2	-
DRAW_ARROW	Viene disegnato come simboli (possono avere valori vuoti)	1	-
DRAW_ZIGZAG	Simile allo stile DRAW_SECTION, ma a differenza di esso, può tracciare segmenti verticali su una barra	2	-
DRAW_FILLING	Riempimento colore tra due linee. 2 valori dei buffer corrispondenti sono	2	-

Plotting	Descrizione	Valore buffer	Color buffers
	mostrati nella Finestra Dati		
DRAW_BARS	Viene disegnato come barre. 4 valori dei buffer corrispondenti vengono visualizzati nella Finestra Dati	4	-
DRAW_CANDLES	Disegnato come Candele giapponesi. 4 valori dei buffer corrispondenti vengono visualizzati nella Finestra Dati	4	-
DRAW_COLOR_LINE	Una linea per la quale è possibile alternare i colori su diverse barre o cambiare il suo colore in qualsiasi momento	1	1

Plotting	Descrizione	Valore buffer	Color buffers
DRAW_COLOR_SECTION	Simile allo stile DRAW_SECTION, ma il colore di ogni sezione può essere impostato singolarmente; il colore può anche essere impostato dinamicamente	1	1
DRAW_COLOR_HISTOGRAM	Simile allo stile DRAW_HISTOGRAM, ma ogni striscia può avere un colore diverso, è possibile impostare il colore in modo dinamico	1	1
DRAW_COLOR_HISTOGRAM2	Simile allo stile DRAW_HISTOGRAM2, ma ogni striscia	2	1

Plotting	Descrizione	Valore buffer	Color buffers
	può avere un colore diverso, è possibile impostare il colore in modo dinamico		
DRAW_COLOR_ARROW	Simile allo stile DRAW_ARROW, ma ogni simbolo può avere il suo colore. Il colore può essere modificato dinamicamente	1	1
DRAW_COLOR_ZIGZAG	Lo stile DRAW_ZIGZAG con le opzioni di colorazione individuale di sezioni e cambiamento dinamico del colore	2	1
DRAW_COLOR_BARS	Lo stile DRAW_B	4	1

Plotting	Descrizione	Valore buffer	Color buffers
	ARS con le opzioni di colorazione individuale delle barre e cambiamento dinamico del colore		
DRAW_COLOR_CANDLES	Lo stile DRAW_CANDLES con le opzioni di colorazione individuale delle candele e colori dinamici che cambiano	4	1

La differenza tra un buffer indicatore ed un array

In ogni indicatore, sul suo livello globale, si dovrebbe dichiarare uno o più array di tipo double, che poi deve essere usato come un buffer utilizzando la funzione dell'indicatore SetIndexBuffer(). Per disegnare i plots indicatore, vengono utilizzati solo i valori dei buffer di indicatore, eventuali altri array non possono essere utilizzati per questo scopo. Inoltre, i valori del buffer vengono visualizzati nella finestra dei dati.

Un buffer indicatore deve essere dinamico e non richiede specificazione della grandezza - la grandezza dell'array utilizzato come buffer indicatore, viene impostato dal sottosistema di esecuzione del terminale, automaticamente

Dopo che l'array è vincolato al buffer indicatore, la direzione di indicizzazione viene impostata di default come negli array ordinari, ma è possibile utilizzare la funzione ArraySetAsSeries() per cambiare

il modo di accesso agli elementi di un array. Per impostazione predefinita, il buffer indicatore viene utilizzato per memorizzare i dati utilizzati per la stampa (INDICATOR_DATA).

Se il calcolo dei valori degli indicatori richiede il possesso di calcoli intermedi e memorizza i valori aggiuntivi per ogni barra, allora un tale array può essere dichiarato come un buffer di calcolo durante il binding (INDICATOR_CALCULATIONS). Per i valori intermedi, è anche possibile utilizzare un array regolare, ma in questo caso, il programmatore deve gestire la grandezza dell'array.

Alcuni plots permettono di impostare un colore per ogni barra. Per memorizzare le informazioni sul colore, vengono utilizzati i buffers colore (INDICATOR_COLOR_INDEX). Il colore è di tipo intero color, ma tutti i buffer degli indicatori devono essere di tipo double. I valori di colore e buffers ausiliari (INDICATOR_CALCULATIONS) non possono essere ottenuti usando CopyBuffer().

Il numero di buffers indicatore deve essere specificato utilizzando la direttiva del compilatore #property indicator_buffers numero_di_buffers:

```
#property indicator_buffers 3  //  1'indicatore ha 3 buffers
```

Il numero massimo consentito di buffers in un indicatore è 512.

Rilevanza del Buffer Indicatore e Plotting

Ogni plotting è basata su uno o più buffer indicatore. Così, per la visualizzazione di candele semplici, sono necessari quattro valori - i prezzi Open, High, Low e Close. Di conseguenza, per visualizzare un indicatore in forma di candele, è necessario dichiarare 4 buffer indicatore e 4 array di tipo double per essi. Ad esempio:

```
//--- L'indicatore ha quattro buffer di indicatori
#property indicator_buffers 4
//--- L'indicatore ha un plotting
#property indicator_plots   1
//--- Il numero del plotting grafico 1 apparirà come candele
#property indicator_type1   DRAW_CANDLES
//--- Le candele saranno disegnate in clrDodgerBlue
#property indicator_color1  clrDodgerBlue
//--- 4 array per i buffer di indicatore
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

I plots grafici utilizzano automaticamente i buffer indicatore in base al numero di plot. La numerazione dei plots trame inizia con 1, la numerazione dei buffer inizia con zero. Se il primo plotting richiede 4 buffer indicatore, allora i primi 4 buffer indicatore saranno utilizzati per disegnarla. Questi quattro buffer dovrebbero collegati con gli array appropriati con l'indicizzazione corretta utilizzando la funzione SetIndexBuffer().

```
//--- Array con associazione ai buffers indicatore
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // Il primo buffer corrisponde all'indic
```

```
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // Il secondo buffer corrisponde all'ind
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // Il terzo buffer corrisponde all'indi
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // Il quarto buffer corrisponde all'ind
```

Per le candele disegnate, l'indicatore le utilizzerà solo i primi quattro buffer, perché il plotting delle "candele" è stato annunciato con il primo numero.

Modifichiamo l'esempio, e aggiungiamo un plotting di una linea semplice - DRAW_LINE. Supponiamo ora che la linea è numerata 1, e le candele sono numero 2. Il numero di buffer ed il numero di plots è aumentato.

```
//--- L'indicatore è dotato di 5 buffer indicatore
#property indicator_buffers 5
//--- L'indicatore ha 2 plots
#property indicator_plots   2
//--- Plot 1 è una linea
#property indicator_type1   DRAW_LINE
//--- Il colore della linea è clrDodgerRed
#property indicator_color1  clrDodgerRed
//--- Plot 2 viene disegnato come candele Giapponesi
#property indicator_type2   DRAW_CANDLES
//--- Il colore delle candele è clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- 5 array per i buffer indicatore
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

L'ordine degli dei plots è cambiato, ed ora la linea viene prima, seguita dalle candele giapponesi. Pertanto, l'ordine dei buffer è appropriato - prima segnaliamo un buffer per la linea con l'indice pari a zero, e poi quattro buffer per le candele.

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // Il primo buffer corrisponde all'in
//--- Array associante con buffer di indicatori per le candele
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // Il secondo buffer corrisponde all
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // Il terzo buffer corrisponde all'in
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // Il quarto buffer corrisponde all'
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // Il quinto buffer corrisponde all'
```

Il numero di buffers e plots può essere impostato solo utilizzando direttive del compilatore, è impossibile modificare queste proprietà in modo dinamico utilizzando le funzioni.

Color Versions of Styles

Come si può vedere dalla tabella, gli stili sono divisi in due gruppi. Il primo gruppo comprende gli stili in cui nel nome non c'è una parola **COLOR**, noi chiamiamo questi stili di base:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

Nel secondo gruppo, i nomi di stile comprendono il termine **COLOR**, chiamiamole versioni colore:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

Tutte le versioni di colore degli stili differiscono da quelli di base, in quanto permettono di specificare un colore per ogni parte del plotting. La minima parte del plotting è una barra, così possiamo dire dire che le versioni colore permettono di impostare il colore su ogni barra.

Eccezioni sono stili DRAW_NONE e DRAW_FILLING, non hanno versioni di colore.

Per impostare il colore tracciando su ciascuna barra, un buffer aggiuntivo per memorizzare l'indice del colore è stato aggiunto alla versione colori. Questi indici indicano il numero di un colore in un array speciale, che contiene un insieme predefinito di colori. La dimensione dell'array di colori è 64. Ciò significa che ogni versione colore di uno stile permette di disegnare un plot in 64 colori diversi.

Il set e il numero di colori nell' array speciale di colori, possono essere impostati tramite una direttiva del compilatore #proprietà indicator_color, dove è possibile specificati tutti i colori necessari separati da virgole. Ad esempio, tale iscrizione in un indicatore:

```
//--- Definisce 8 colori per colorare le candele (sono memorizzati nell'array speciale
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
```

Si afferma che per tracciarne 1, 8 colori sono impostati, che saranno collocati in un array speciale. Ulteriormente nel programma non specificheremo il colore del tracciato, ma solo il suo indice. Se vogliamo impostare il colore rosso per il numero di barra **K**, il valore di indice di colore da un array deve essere impostato nel buffer colore dell'indicatore Il colore rosso è specificato prima della direttiva, corrisponde al numero 0 dell' indice.

```
  //--- imposta il colore della candela clrRed
  col_buffer[buffer_index]=0;
```

L'insieme dei colori non è dato una volta per tutte, può essere variato dinamicamente utilizzando PlotIndexSetInteger(). Esempio:

```
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scriviar
                          color_array[i]);      //  Un nuovo colore
```

Proprietà del l'indicatore e plotting

Per i plots indicatore, le proprietà possono essere impostate mediante direttive del compilatore ed utilizzando le funzioni appropriate. Per sapere più informazioni su questo in: Collegamento tra Proprietà Indicatore e Funzioni. Il cambiamento dinamico delle proprietà di indicatori utilizzando funzioni speciali, permette di creare indicatori personalizzati più flessibili.

Inizio del Disegno Indicatore sul Grafico

In molti casi, a seconda delle condizioni dell'algoritmo, è impossibile avviare il calcolo dei valori degli indicatori immediatamente con la barra di corrente, in quanto è necessario prevedere un numero minimo di barre precedenti disponibili nello storico. Ad esempio, molti tipi di smoothing implicano l'utilizzo di una matrice di prezzi nelle precedenti N barre, e sulla base di questi valori, viene calcolato il valore indicatore sulla barra corrente.

In tali casi, non esiste alcun modo per calcolare i valori degli indicatori per le prime N barre, o questi valori non sono destinati ad essere visualizzati sul grafico e sono solo sussidiari al calcolo di ulteriori valori. Per evitare il plotting dell'indicatore sulle prime N barre dello storico, impostare il valore di N per la proprietà PLOT_DRAW_BEGIN per il plot corrispondente:

```
//--- Array associante con buffer di indicatori per le candele
PlotIndexSetInteger(number_of_plot,PLOT_DRAW_BEGIN,N);
```

Qui:

- numero_di_plot- un valore da 2zero a indicator_plots-1 (la numerazione dei plots inizia con zero).
- N - il numero delle prime barre nello storico, in cui l'indicatore non deve essere visualizzato sul grafico.

DRAW_NONE

Lo stile DRAW_NONE è progettato per l'utilizzo in casi in cui è necessario calcolare i valori di un buffer e mostrarli nella Finestra Dati, ma il tracciamento sul grafico non è necessario. Per impostare la precisione usare l'espressione IndicatorSetInteger (INDICATOR_DIGITS, num_chars) nella funzione OnInit():

```
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- Imposta la precisione dei valori da visualizzare nella finestra dei dati
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

Il numero di buffer necessari per tracciare DRAW_NONE è 1.

Un esempio di indicatore che mostra il numero della barra su cui il mouse attualmente si muove, nella Finerstra Dati. La numerazione corrisponde alle timeseries, ovvero che l'attuale barra incompiuta ha l'indice pari zero, e la barra più vecchia ha l'indice più grande.

Si noti che nonostante il fatto che per il colore rosso è impostato il plotting **#1**, l'indicatore non disegna nulla sul grafico.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_NONE.mq5  |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

```
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Invisible
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- buffers indicatore
double         InvisibleBuffer[];
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- Associa un array e un buffer indicatore
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- Imposta la precisione dei valori da visualizzare nella finestra dei dati
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static datetime lastbar=0;
//--- Se questo è il primo calcolo dell'indicatore
   if(prev_calculated==0)
     {
      //--- Rinumera le barre per la prima volta
      CalcValues(rates_total,close);
      //--- Ricorda il tempo di apertura della barra corrente in lastbar
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
```

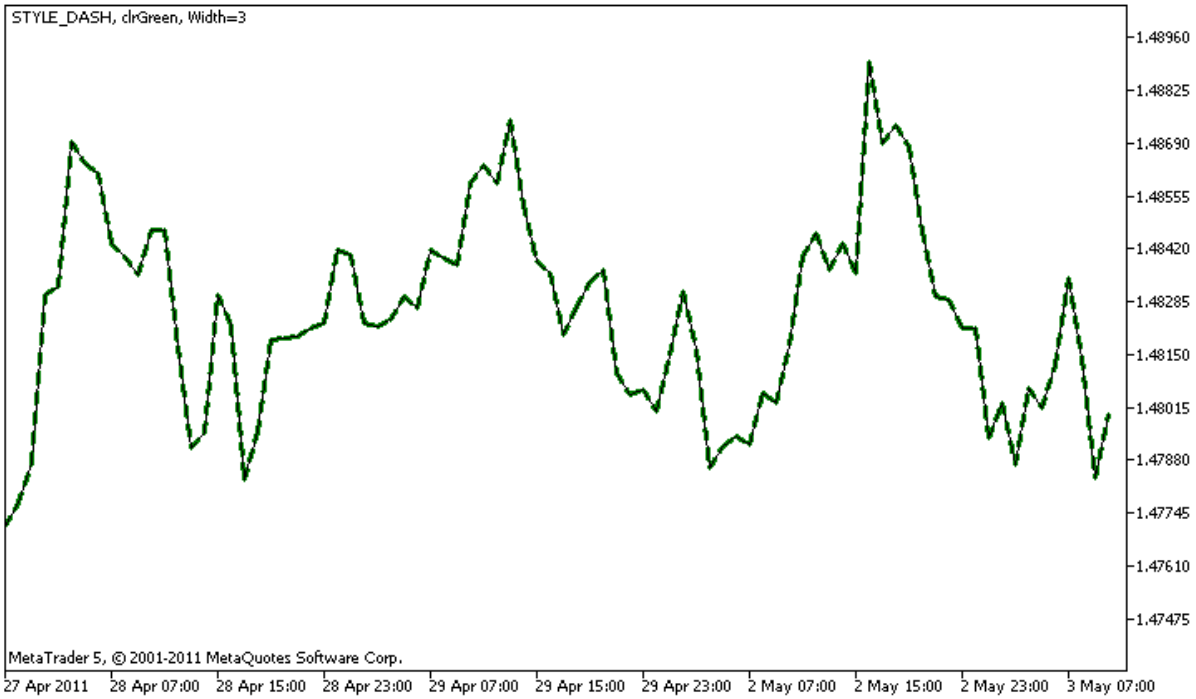
```
      }
    else
      {
       //--- Se una nuova barra è apparsa, il suo tempo di apertura differisce da lastb
       if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
         {
          //--- Rinumera le barre ancora una volta
          CalcValues(rates_total,close);
          //--- Aggiornare l'orario di apertura della barra corrente in lastbar
          lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
         }
      }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Numera le barre come in una timeseries                                             |
//+------------------------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- Imposta l' indicizzazione del buffer indicatore come in una timeseries
   ArraySetAsSeries(InvisibleBuffer,true);
//--- Riempie ogni barra con il suo numero
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

DRAW_LINE

DRAW_LINE traccia una linea del colore specificato dai valori del buffer dell'indicatore. La larghezza, lo stile e il colore della linea possono essere impostati utilizzando le direttive del compilatore e dinamicamente utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambia a seconda della situazione attuale.

Il numero di buffer necessari per tracciare DRAW_LINE è 1.

Un esempio di indicatore che disegna una linea utilizzando i prezzi Close delle barre. Il colore, spessore e stile cambia casualmente ogni n=5 ticks.



Notare che inizialmente per **Plot1** con DRAW_LINE le proprietà vengono impostate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() queste tre proprietà vengono impostate in modo casuale. Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                            DRAW_LINE.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_LINE"
#property description "Disegna una linea di un colore specificato ai prezzi Close"
#property description "Colore, spessore e stile della linea vengono cambiati casualmer
#property description "dopo ogni N ticks"
```

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Le proprietà della linea vengono impostate utilizzando le direttive del compilat
#property indicator_label1  "Line"      // Nome del disegno per la Finestra Dati
#property indicator_type1   DRAW_LINE   // Tipo di disegno della linea
#property indicator_color1  clrRed      // Colore della linea
#property indicator_style1  STYLE_SOLID // Stile della linea
#property indicator_width1  1           // Spessore della linea
//--- parametri input
input int      N=5;          // Numero di ticks da cambiare
//--- Un buffer indicatore per il disegno
double         LineBuffer[];
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen};
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- Associa un array e un buffer indicatore
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- Inizializzazione del generatore di numeri pseudo-casuali
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
```

```
      {
       //--- Cambia le proprietà della linea
       ChangeLineAppearance();
       //--- Resetta il contatore dei ticks a zero
       ticks=0;
      }

//--- Blocco per il calcolo dei valori dell'indicatore
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Modifica l'aspetto della linea disegnata nell'indicatore                           |
//+------------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco per cambiare il colore della linea
//--- Ottiene un numero casuale
   int number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore della linea
   comm=comm+(string)colors[color_index];

//--- Un blocco per modificare la larghezza della linea
   number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5; // Lo spessore è impostato da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+", Width="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
```

```
   int style_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_SECTION

DRAW_SECTION disegna sezioni del colore specificato dai valori del buffer dell'indicatore. La larghezza, il colore e lo stile della linea possono essere specificati come per lo stile DRAW_LINE - utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambi a seconda della situazione attuale.

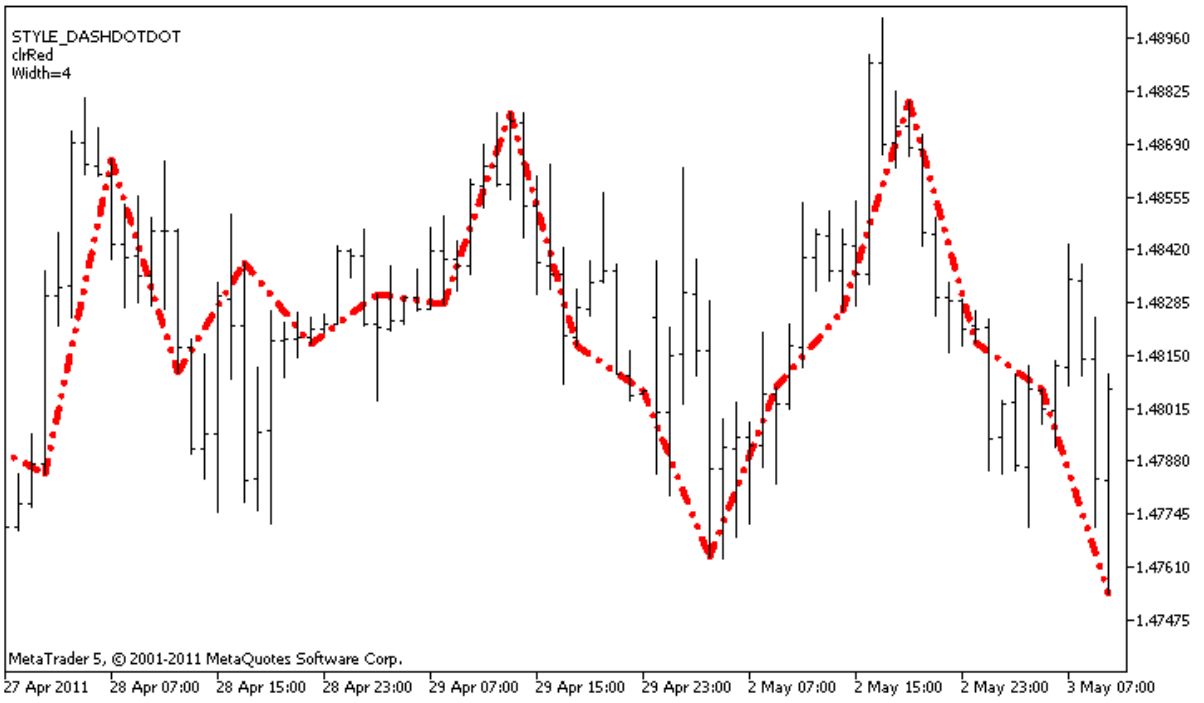
Le sezioni vengono disegnate da un valore non vuoto di un altro valore non vuoto del buffer indicatore, i valori vuoti vengono ignorati. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nella proprietà PLOT_EMPTY_VALUE: ad esempio, se l'indicatore deve essere disegnato come una sequenza di sezioni su valori diversi da zero, allora è necessario impostare il valore zero come uno vuoto:

```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Riempire sempre esplicitamente i valori dei buffer indicatore, impostare un valore vuoto in un buffer per gli elementi che non devono essere tracciati.

Il numero di buffer necessari per tracciare DRAW_SECTION è 1.

Un esempio di indicatore che disegna sezioni tra i prezzi High e Low. Il colore, la larghezza e lo stile di **tutte** le sezioni cambiano in modo casuale ogni **N** ticks.



Notare che inizialmente per **Plot1** con DRAW_SECTION le proprietà vengono impostate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() queste tre proprietà vengono impostate in modo casuale. Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------+
//|                                                DRAW_SECTION.mq5 |
```

```
//|                                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                                 https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_SECTION"
#property description "Disegna sezioni dritte ogni N barre"
#property description "Il colore, spessore e stile delle sezioni vengono cambiati in r
#property description "dopo ogni N ticks"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Section
#property indicator_label1  "Section"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri input
input int      bars=5;           // La lunghezza della sezione in barre
input int      N=5;              // Il numero di ticks per cambiare lo stile delle sez
//--- Un buffer indicatore per il disegno
double         SectionBuffer[];
//--- Una variabile ausiliaria per calcolare la fine delle sezioni
int            divider;
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen};
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- Associa un array e un buffer indicatore
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Controllare il parametro indicatore
   if(bars<=0)
     {
      PrintFormat("Valore non valido del parametro barra=%d",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
//---+
```

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Il numero della barra da cui inizia il calcolo dei valori degli indicatori
   int start=0;
//--- Se l'indicatore è stato calcolato prima, allora impostare l'inizio sulla barra p
   if(prev_calculated>0) start=prev_calculated-1;
//--- Qui ci sono tutti i calcoli dei valori degli indicatori
   for(int i=start;i<rates_total;i++)
     {
/ / --- Ottiene il resto della divisione del numero di barre di 2*barre
      int rest=i%divider;
      //--- Se il numero di barre è divisibile per 2*barre
      if(rest==0)
        {
         //--- Imposta la fine della sezione presso il prezzo elevato di questo bar
         SectionBuffer[i]=high[i];
        }
      //--- Se il resto della divisione è pari a barre,
      else
        {
         //--- Imposta la fine della sezione presso il prezzo elevato di questo bar
         if(rest==bars) SectionBuffer[i]=low[i];
         //--- Se non è successo niente, ignora bar - set 0
```

```
         else SectionBuffer[i]=0;
        }
      }
//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Modifica l'aspetto delle sezioni dell'indicatore                                   |
//+------------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco del cambiamento del colore della linea
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore della linea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Un blocco per modificare la larghezza della linea
   number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La larghezza è impostata da 0 a 4
//--- Imposta la larghezza
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta lo stile della linea
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

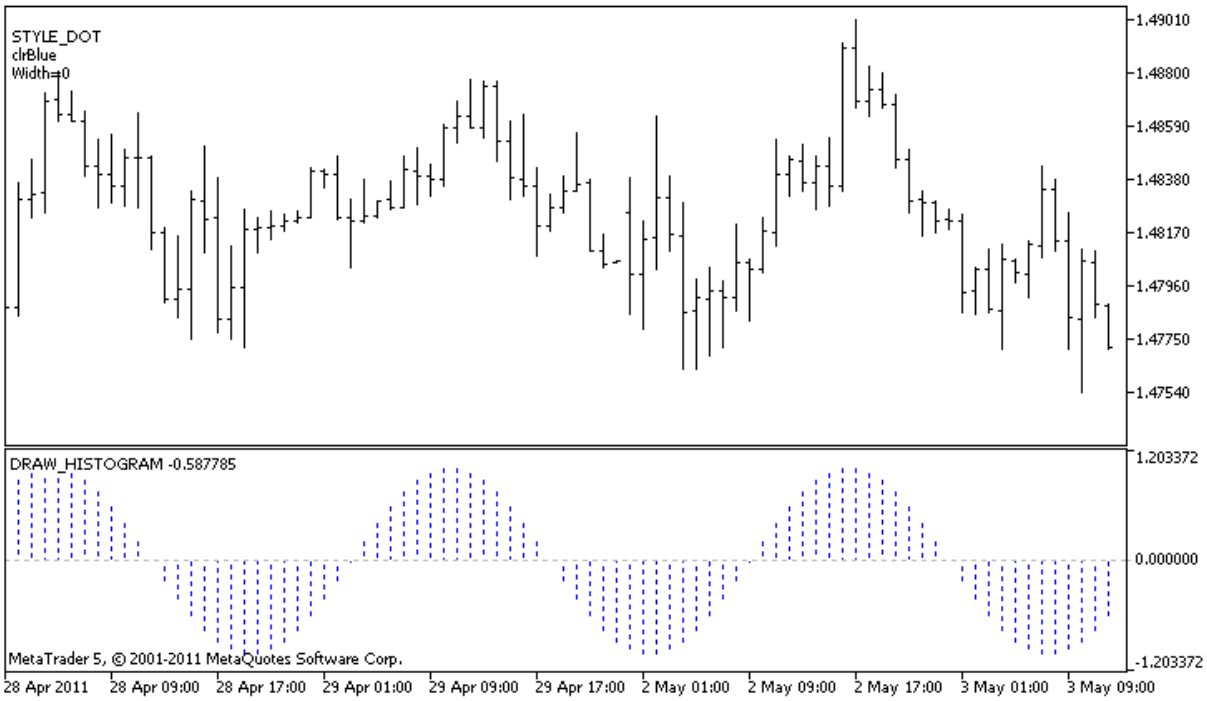
DRAW_HISTOGRAM

Lo stile DRAW_HISTOGRAM disegna un istogramma come una sequenza di colonne di un colore specificato da zero ad un valore specificato. I valori sono presi dal buffer indicatore. La larghezza, il colore e lo stile della colonna possono essere specificati come per lo stile DRAW_LINE - utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di cambiare l'aspetto dell'istogramma sulla base della situazione attuale.

Dal momento che una colonna dal livello zero è disegnata su ogni barra, DRAW_HISTOGRAM dovrebbe meglio essere utilizzata in una finestra di grafico separata. Molto spesso questo tipo di tracciamento viene utilizzato per creare indicatori del tipo oscillatore, per esempio, Bears Power oppure OSMA. Per i valori vuoti non visualizzabili deve essere specificato il valore zero.

Il numero di buffer necessari per tracciare DRAW_HISTOGRAM è 1.

Un esempio di indicatore che disegna una sinusoide di un colore specificato basato sulla funzione MathSin(). Il colore, la larghezza e lo stile di **tutte** le colonne dell'istogramma cambiano in modo casuale ogni **N** ticks. Il parametro barre specifica il periodo della sinusoide, che è dopo il numero specificato di barre per cui la sinusoide ripeterà il ciclo.



Notare che inizialmente per **Plot1** con DRAW_HISTOGRAM le proprietà vengono impostate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() queste tre proprietà vengono impostate in modo casuale. Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_HISTOGRAM"
#property description "Esso disegna una sinusoide come un istogramma in una finestra s
#property description "Il colore e la larghezza delle colonne vengono modificate in mo
#property description "dopo ogni N ticks"
#property description "I parametri delle barre impostano il numero delle barre nel cic

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      bars=30;          // Il periodo della sinusoide in barre
input int      N=5;              // Il numero di ticks che cambiano l'istogramma
//--- buffers indicatore
double         HistogramBuffer[];
//--- Un fattore per ottenere l'angolo di 2Pi in radianti, moltiplicato per il paramet
double    multiplier;
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen};
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato  |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- Calcola il moltiplicatore
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Imposta il valore di bars=%d maggiore di 1",barre);
      //--- Terminazione precoce dell'indicatore
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione di iterazione indicatore personalizato |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Calcola i valori dell'indicatore
   int start=0;
//--- Se già calcolato nelle precedenti starts di OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // imposta l'inizio del calcolo con
//--- Compila il buffer di indicatore con valori
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Cambia l'apparenza delle linee nell'indicatore                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco per cambiare il colore della linea
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
```

```
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore della linea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Un blocco per modificare la larghezza della linea
   number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La larghezza è impostata da 0 a 4
//--- Imposta la larghezza
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta lo stile della linea
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_HISTOGRAM2

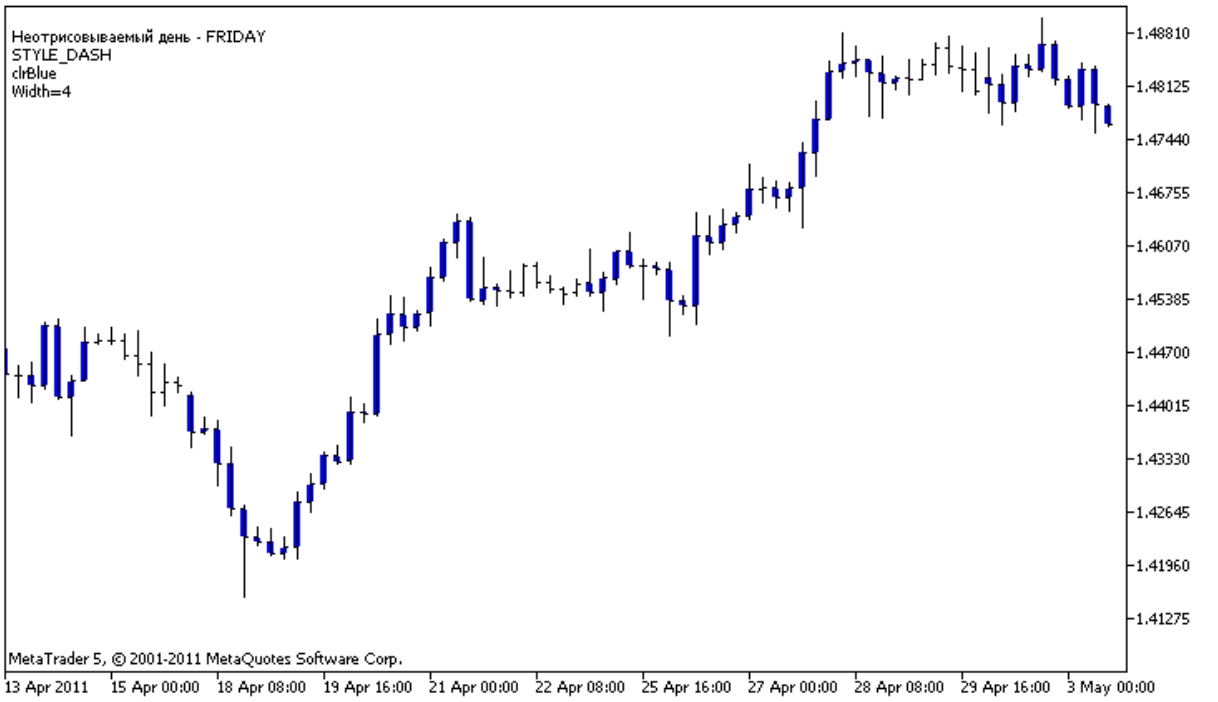
Lo stile DRAW_HISTOGRAM2 disegna un istogramma di un colore specificato - segmenti verticali utilizzano i valori di due buffer indicatore. La larghezza, il colore e lo stile dei segmenti possono essere specificati come per lo stile DRAW_LINE - utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di cambiare l'aspetto dell'istogramma sulla base della situazione attuale.

Lo stile DRAW_HISTOGRAM2 può essere utilizzato in una sottofinestra separata di un grafico e nella sua finestra principale. Per i valori vuoti nulla viene tracciato, tutti i valori nei buffer indicatore devono essere impostati in modo esplicito. I buffer non vengono inizializzati con un valore zero.

Il numero di buffer necessari per tracciare DRAW_HISTOGRAM2 è 2.

Un esempio di indicatore che traccia un segmento verticale del colore specificato e la larghezza tra i prezzi Open e Close di ogni barra. Il colore, la larghezza e lo stile di **tutte** le colonne dell'istogramma cambiano in modo casuale ogni **N** ticks. Durante l'inizio dell'indicatore, nella funzione OnInit(), il numero del giorno della settimana per il quale non sarà disegnato l'istogramma - invisible_day - è impostato in modo casuale. A questo scopo viene impostato un valore vuoto PLOT_EMPTY_VALUE=0:

```
//--- Imposta un valore vuoto
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```



Notare che inizialmente per **Plot1** con DRAW_HISTOGRAM2 le proprietà vengono impostate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() queste tre proprietà sono impostate in modo casuale. Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                    DRAW_HISTOGRAM2.mq5 |
//|                                 Copyright 2011, MetaQuotes Software Corp. |
//|                                                    https://www.mql5.com |
```

```
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_HISTOGRAM2"
#property description "Disegna un segmento tra Open e Close su ciascuna barra"
#property description "Il colore, lo spessore e lo stile vengono cambiati casualmente"
#property description "dopo ogni N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Histogram_2
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      N=5;              // Il numero di ticks che cambiano l'istogramma
//--- buffers indicatore
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- Il giorno della settimana per il quale l'indicatore non viene tracciato
int invisible_day;
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen};
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- Imposta un valore vuoto
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Ottiene un numero casuale da 0 a 5
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

```
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Calcola i valori dell'indicatore
   int start=0;
//--- Per ottenere il giorno della settimana per il prezzo di apertura di ogni barra
   MqlDateTime dt;
//--- Se già calcolato nelle precedenti starts di OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // imposta l'inizio del calcolo con
//--- Compila il buffer di indicatore con valori
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      if(dt.day_of_week==invisible_day)
        {
         Histogram_2Buffer1[i]=0;
         Histogram_2Buffer2[i]=0;
        }
      else
        {
         Histogram_2Buffer1[i]=open[i];
         Histogram_2Buffer2[i]=close[i];
        }
     }
//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Cambia l'apparenza delle linee nell'indicatore                               |
//+------------------------------------------------------------------------------+
```

```
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco del cambiamento del colore della linea
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore della linea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Un blocco per modificare la larghezza della linea
   number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La larghezza è impostata da 0 a 4
//--- Imposta lo spessore della linea
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta lo stile della linea
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Aggiunge le informazioni sul giorno che viene omesso nei calcoli
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_ARROW

Lo stile DRAW_ARROW disegna frecce del colore specificato (simboli del set Wingdings) in base al valore del buffer dell'indicatore. La larghezza ed il colore dei simboli può essere specificata come per lo stile DRAW_LINE - utilizzando le direttive del compilatore o in modo dinamico con la funzione PlotIndexSetInteger(). I cambiamenti dinamici delle proprietà di plottaggio permettono di cambiare l'aspetto di un indicatore sulla base della situazione attuale.

Il codice simbolo viene impostato utilizzando la proprietà PLOT_ARROW

```
//--- Definisce il codice simbolo dal carattere Wingdings per disegnare in PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Il valore predefinito di PLOT_ARROW=159 (un cerchio).

Ogni freccia è in realtà un simbolo che ha l'altezza ed il punto di ancoraggio, e può coprire alcune importanti informazioni su un grafico (per esempio, il prezzo di chiusura alla bar). Quindi, possiamo anche specificare lo slittamento verticale in pixel, che non dipende dalla scala del grafico. Le frecce saranno spostate verso il basso per il numero di pixel specificato, anche se i valori dell'indicatore rimarranno gli stessi:

```
//--- Imposta lo spostamento verticale di frecce in pixel
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

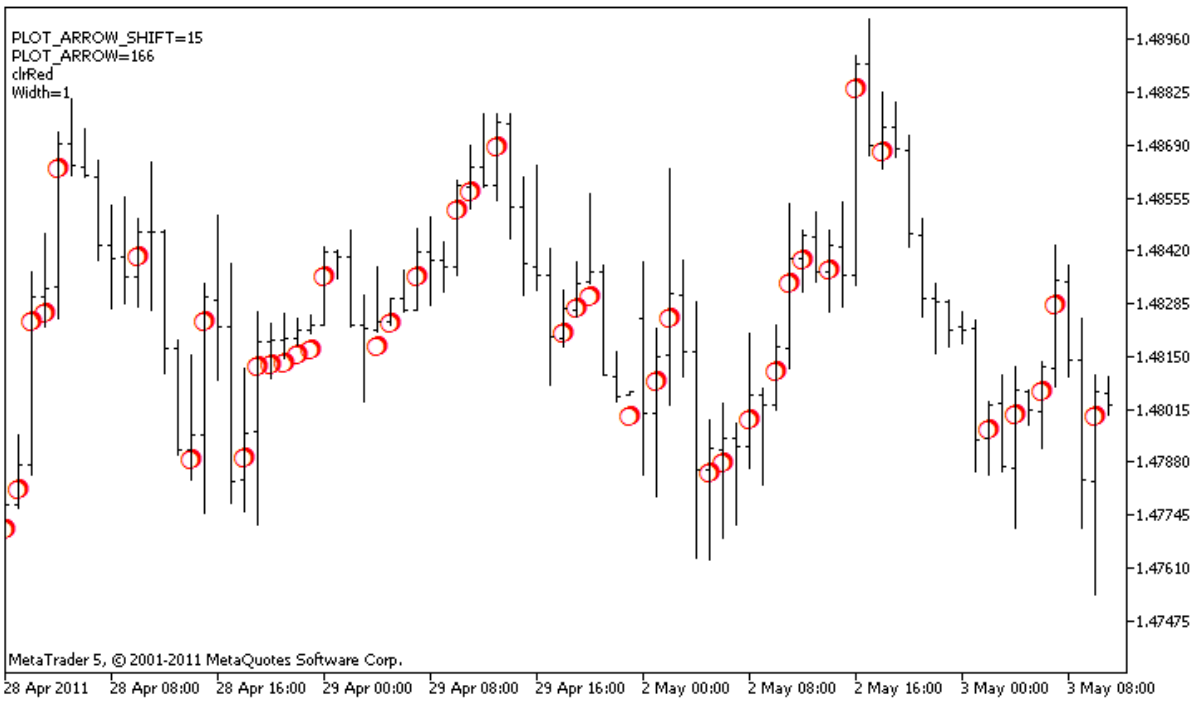
Un valore negativo di PLOT_ARROW_SHIFT significa lo spostamento delle frecce verso l'alto, un valore positivo slitta la freccia verso il basso.

Lo stile DRAW_ARROW può essere utilizzato in una sottofinestra separata di un grafico e nella sua finestra principale. Valori vuoti non vengono disegnati e non compaiono nella "Finestra Dati"; tutti i valori dei buffer indicatori devono essere impostati in modo esplicito. I buffer non vengono inizializzati con un valore zero.

```
//--- Imposta un valore vuoto
   PlotIndexSetDouble(index_of_plot_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

Il numero di buffer necessari per il plotting di DRAW_ARROW è 1.

Un esempio di indicatore, che disegna frecce su ciascuna barra con il prezzo close superiore al prezzo di chiusura della barra precedente. Il colore, spessore, slittamento e codice simbolo di **tutte** le frecce vengono cambiati in modo casuale ogni **N** ticks.



Nell'esempio, per **Plot1** con lo stile DRAW_ARROW, la proprietà, il colore e la grandezza vengono specificati utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() le proprietà sono impostate in modo casuale. Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                            DRAW_ARROW.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per mostrare DRAW_ARROW"
#property description "Disegna frecce impostate da caratteri Unicode, su un grafico"
#property description "Il colore, grandezza, slittamento e codice del smbolo vengono c
#property description "dopo ogni N ticks"
#property description "Il parametro code imposta il valore base: code=159 (un cerchio)

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- disegna Frecce
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- parametri di input
```

```
input int      N=5;          // Numero di ticks da cambiare
input ushort   code=159;     // Codice simbolo da disegnare in DRAW_ARROW
//--- Un buffer indicatore per il disegno
double         ArrowsBuffer[];
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- Definisce il codice simbolo per il disegnamento in PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- Imposta lo spostamento verticale di frecce in pixel
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- Imposta come un valore vuoto 0
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare il colore, la grandezza , slittamento e codice dell
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Blocco per il calcolo dei valori dell'indicatore
```

```
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- Ciclo del calcolo
   for(int i=1;i<rates_total;i++)
     {
      //--- Se il prezzo Close corrente è maggiore di quello precedente, disegna una
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- Altrimenti specifica il valore zero
      else
         ArrowsBuffer[i]=0;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Modifica l'aspetto dei simboli nell' indicatore                              |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà degli indicatori
   string comm="";
//--- Un blocco per il cambio del colore della freccia
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore della linea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Un blocco per cambiare la dimensione delle frecce
   number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La grandezza è impostata da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive la grandezza della freccia
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Un blocco per il cambio del codice della freccia (PLOT_ARROW)
   number=MathRand();
//--- Ottiene il resto della divisione intera per calcolare un nuovo codice della fre
   int code_add=number%20;
//--- Imposta il nuovo codice simbolo come il risultato di code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- Scrive il codice simbolo PLOT_ARROW
```

```
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- Un blocco per cambiare lo slittamento verticale delle frecce in pixel
   number=MathRand();
//--- Prende lo slittamento come il resto della divisione intera
   int shift=20-number%41;
//--- Imposta il nuovo slittamento da -20 a 20
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- Scrive lo slittamento di PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_ZIGZAG

Lo stile DRAW_ZIGZAG disegna segmenti di un colore specificato in base ai valori dei due buffer indicatori. Questo stile è molto simile a DRAW_SECTION, ma a differenza di quest'ultimo, permette di disegnare segmenti verticali all'interno di una barra, se i valori di entrambi i buffer indicatori sono impostati per questa barra. I segmenti vengono tracciati da un valore nel primo buffer di un valore nel secondo buffer indicatore. Nessuno dei buffer può contenere solo valori vuoti, perché in questo caso nulla è tracciato.

La larghezza, il colore e lo stile della linea possono essere specificati come per lo stile DRAW_SECTION - con le direttive del compilatore o dinamicamente utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambi a seconda della situazione attuale.

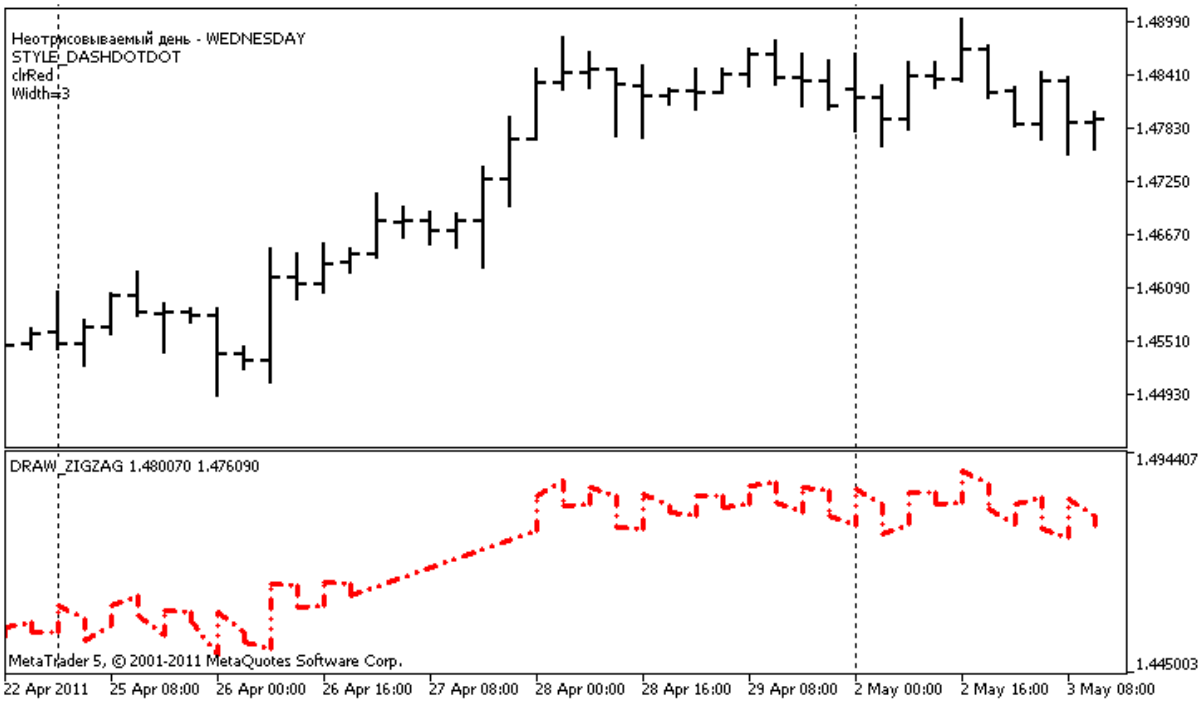
Le sezioni vengono disegnate da un valore non vuoto di un buffer ad un valore non vuoto di un altro buffer indicatore. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nella proprietà PLOT_EMPTY_VALUE:

```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Riempie esplicitamente i valori dei buffer indicatori, imposta un valore vuoto in un buffer per saltare le barre.

Il numero di buffer necessari per tracciare DRAW_ZIGZAG è 2.

Un esempio di indicatore che traccia una sega sulla base dei prezzi High e Low. Il colore, la larghezza e lo stile delle linee a zig-zag cambiano in modo casuale ogni **N** ticks.



Notare che inizialmente per **Plot1** con DRAW_ZIGZAG le proprietà vengono impostate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() queste proprietà vengono

impostate in modo casuale. Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                              DRAW_ZIGZAG.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_ZIGZAG"
#property description "Esso disegna una sega come segmenti dritti, saltando le barre c
#property description "Il giorno da saltare è selezionato a caso durante l'avvio dell'
#property description "Il colore, spessore e stile dei segmenti vengono cambiati in mo
#property description " ogni N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ZigZag
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      N=5;              // Numero di ticks da cambiare
//--- buffers indicatore
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- Il giorno della settimana per il quale l'indicatore non viene tracciato
int invisible_day;
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen};
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- Legatura Array e buffer indicatore
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- Ottiene un valore casuale da 0 a 6, per questo giorno l'indicatore non è traccia
   invisible_day=MathRand()%6;
/ / --- Il valore 0 (vuoto) parteciperà al disegno
```

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero sufficiente di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- La struttura del tempo è richiesta per ottenere il giorno della settimana di ogr
   MqlDateTime dt;

//--- La posizione di partenza dei calcoli
   int start=0;
//--- Se l'indicatore è stato calcolato sulla tick precedente, allora avviare il calc
   if(prev_calculated!=0) start=prev_calculated-1;
//--- Ciclo del calcolo
   for(int i=start;i<rates_total;i++)
     {
      //--- Scrive il tempo di apertura della barra nella struttura
      TimeToStruct(time[i],dt);
      //--- Se il giorno della settimana di questa barra è pari ad invisible_day
      if(dt.day_of_week==invisible_day)
        {
         //--- Scrive i valori vuoti per i buffer per questa barra
         ZigZagBuffer1[i]=0;
```

```
         ZigZagBuffer2[i]=0;
        }
      //--- Se il giorno della settimana è ok, riempire i buffer
      else
        {
         //--- Se il numero di barre è pari
         if(i%2==0)
           {
            //---  Write High in the 1st buffer and Low in the 2nd one
            ZigZagBuffer1[i]=high[i];
            ZigZagBuffer2[i]=low[i];
           }
         //--- Il numero di barre è dispari
         else
           {
            //--- Compila le barre in un ordine inverso
            ZigZagBuffer1[i]=low[i];
            ZigZagBuffer2[i]=high[i];
           }
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Modifica l'aspetto dei segmenti a zig-zag                                    |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà ZigZag
   string comm="";
//--- Un blocco per cambiare il colore del ZigZag
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore della linea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Un blocco per modificare la larghezza della linea
   number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La larghezza è impostata da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
```

```
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Aggiunge le informazioni sul giorno che viene omesso nei calcoli
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_FILLING

Lo stile DRAW_FILLING disegna un'area colorata tra i valori dei due buffer indicatore. In realtà, questo stile disegna due linee e riempie lo spazio tra di loro con uno dei due colori specificati. E' usato per la creazione di indicatori che disegnano canali. Nessuno dei buffer può contenere solo valori vuoti, perché in questo caso nulla è tracciato.

È possibile impostare due colori di riempimento:

- il primo colore è usato per le zone in cui i valori del primo buffer sono maggiori dei valori nel secondo buffer indicatore;
- il secondo colore è usato per le aree in cui valori nel secondo buffer sono maggiori dei valori nel primo buffer indicatore.

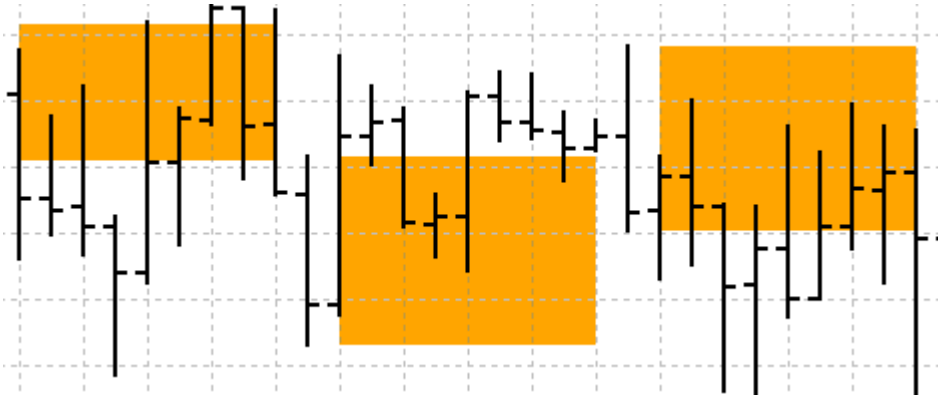
Il colore di riempimento può essere impostato utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambia a seconda della situazione attuale.

L'indicatore è calcolato per tutte le barre, per cui i valori dei due buffer indicatore sono uguali né 0 né al valore vuoto. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nelle proprietà PLOT_EMPTY_VALUE:

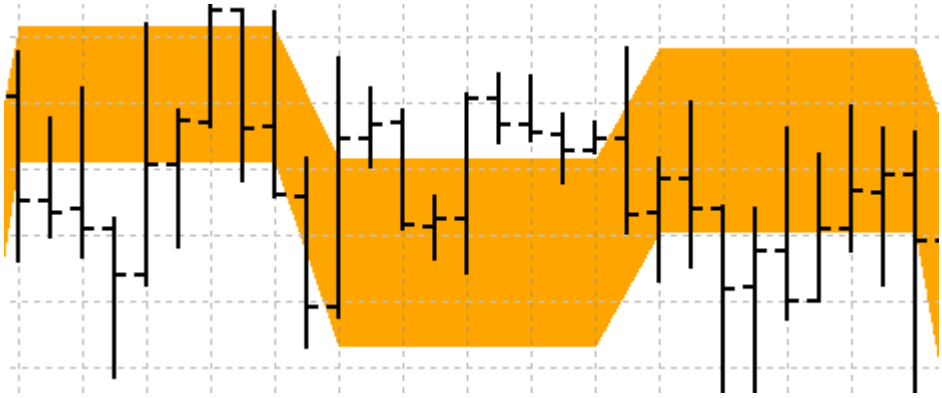
```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- INDICATOR_EMPTY_VALUE (valore vuoto) non partecipa nel calcolo di
   PlotIndexSetDouble (DRAW_FILLING_creation_index,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VA
```

Il disegno delle barre che non partecipano al calcolo dell'indicatore dipende dai valori nei buffer indicatori:

- Le barre, per le quali i valori di entrambi i buffer indicatori sono uguali a 0, non partecipano a disegnare l'indicatore. Ciò significa che l'area con valori zero non viene compilata.



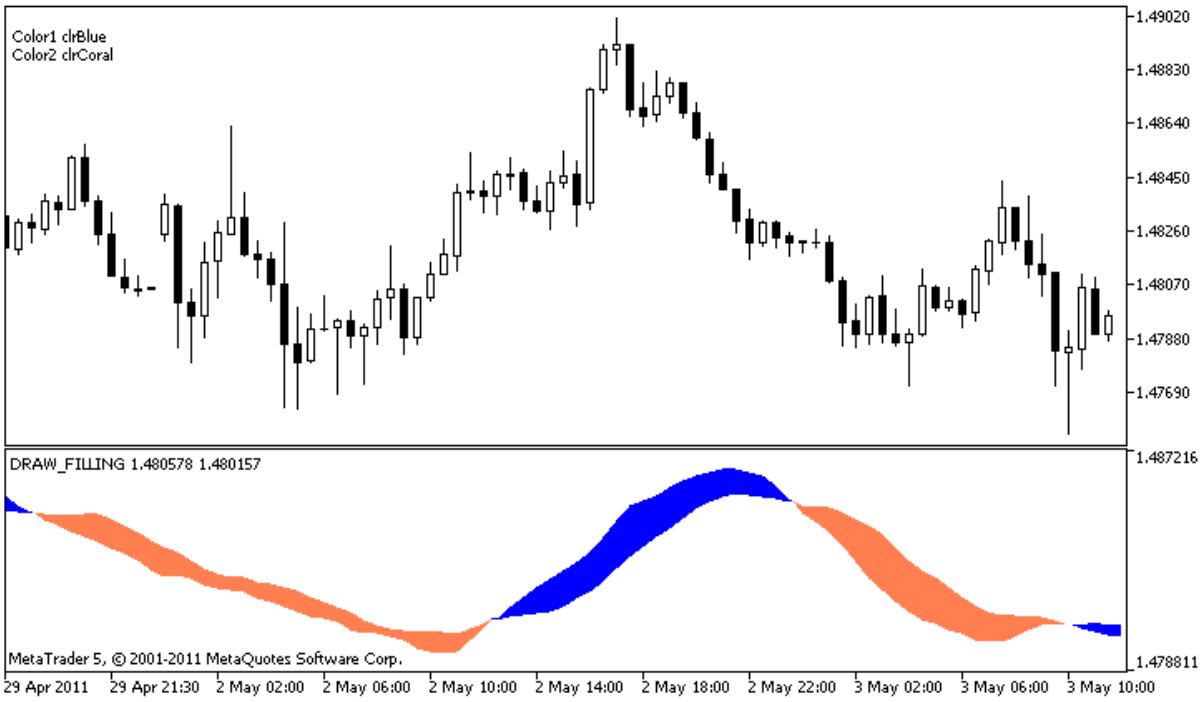
- Barre, per le quali i valori dei buffers indicatori sono pari al "valore vuoto", partecipano a disegnare l'indicatore. La zona con valori vuoti sarà compilata in modo da collegare le aree con valori significativi.



Va notato che se il "valore vuoto" è uguale a zero, le barre che non partecipano nel calcolo dell'indicatore vengono inoltre compilate.

Il numero di buffer necessari per tracciare DRAW_FILLING è 2.

Un esempio di indicatore che disegna un canale tra due MAs con differenti periodi di mediazione in una finestra separata. Il cambiamento dei colori all'incrocio delle medie mobili mostra visivamente il cambiamento delle tendenze al rialzo e al ribasso. I colori cambiano in modo casuale ogni **N** ticks. Il parametro N è impostato nei parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà dell' indicatore).



Notare che inizialmente per **Plot1** con DRAW_FILLING le proprietà vengono impostate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() nuovi colori vengono impostati in modo casuale.

```
//+------------------------------------------------------------------+
//|                                                 DRAW_FILLING.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_FILLING"
#property description "Esso disegna un canale tra due MA in una finestra separata"
#property description "Il colore di riempimento viene cambiato dinamicamente"
#property description "dopo ogni N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Intersection
#property indicator_label1  "Intersezione"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- parametri di input
input int      Fast=13;          // Il periodo di fast MA
input int      Slow=21;          // Il periodo di slow MA
input int      shift=1;          // Lo slittamento dei MA verso il futuro(positivo)
input int      N=5;              // Numero di ticks da cambiare
//--- Buffer Indicatore
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrC
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
   fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
   slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

```
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero sufficiente di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Fa il primo calcolo dell'indicatore, o i dati cambiano e richiede un ricalcolo c
   if(prev_calculated==0)
     {
      //--- Copiare tutti i valori degli indicatori nei buffer appropriati
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // Riempie solo quei dati che sono aggiornati
     {
      //--- Ottiene la differenza nei barre tra l'inizio corrente e precedente di OnCa
      int to_copy=rates_total-prev_calculated;
      //--- Se non c'è differenza, copieremo ancora un valore - sulla barra a zero
      if(to_copy==0) to_copy=1;
      //--- copia i valori to_copy alla fine dei buffer indicatore
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Cambia i colori del riempimento del canale                                   |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
```

```
//--- Un blocco per cambiare il colore della linea
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);

//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index1=number%size;
//--- Imposta il primo colore come proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- Scrive il primo colore
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   number=MathRand(); // Ottiene un numero casuale
   int color_index2=number%size;
//--- Imposta il secondo colore come proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- Scrive il secondo colore
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_BARS

Lo stile DRAW_BARS disegna barre sui valori dei quattro buffer indicatori, che contengono i prezzi Open, High, Low e Close. E' utilizzato per la creazione di indicatori personalizzati come barre, incluse quelli in una sottofinestra separata di un grafico e su altri strumenti finanziari.

Il colore delle barre può essere impostato utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambia a seconda della situazione attuale.

L'indicatore è disegnato solo per queste barre, per cui i valori non vuoti di **tutti** e quattro i buffer indicatore sono impostati. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nelle proprietà PLOT_EMPTY_VALUE:

```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

Riempie esplicitamente i valori dei buffer indicatori, imposta un valore vuoto in un buffer per saltare le barre.

Il numero di buffer richiesti per il plotting di DRAW_BARS è 4. Tutti i buffer per il plotting dovrebbero andare uno dopo l'altro, nell'ordine: Open, High, Low e Close. Nessuno dei buffer può contenere solo valori vuoti, perché in questo caso nulla è tracciato.

Un esempio di indicatore che disegna barre su uno strumento finanziario selezionato in una finestra separata. Il colore delle barre varia casualmente ogni **N** ticks. Il parametro N è impostato nei parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà dell' indicatore).

Si prega di notare che, per **Plot1** con lo stile DRAW_BARS, il colore viene impostato utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() il colore è impostato in modo casuale da una lista preparata in precedenza.

```
//+------------------------------------------------------------------------------+
//|                                                              DRAW_BARS.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_BARS"
#property description "Disegna le barre di un simbolo selezionato in una finestra sepa
#property description "Il colore e lo spessore delle barre, così come il simbolo, venq
#property description "ogni N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "Bars"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      N=5;              // Il numero di ticks per cambiare il tipo
input int      bars=500;         // Il numero di barre da mostrare
input bool     messages=false;   // Mostra i messaggi nel log degli "Expert Advisors"
//--- Buffer Indicatore
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- Nome simbolo
string symbol;
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
/ / --- Se bars è molto piccolo - completa il lavoro prima del tempo
   if(bars<50)
     {
      Comment("Prego specificare un gran numero di barre! L'operazione dell'indicatore
```

```
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mappatura buffers indicatore
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- Il nome del simbolo, per cui le barre vengono disegnate
   symbol=_Symbol;
//--- Imposta il display del simbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- Un valore vuoto
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero sufficiente di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Imposta un nuovo simbolo dalla finestra Market Watch
      symbol=GetRandomSymbolName();
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();

      int tries=0;
      //--- Fa 5 tentativi per riempire il buffer con i prezzi dal simbolo
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- Un contatore delle chiamate della funzione CopyFromSymbolToBuffers()
         tries++;
        }
```

```
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie i buffer indicatore con i prezzi                                           |
//+------------------------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- Nell'array rates[], copieremo Open, High, Low e Close
   MqlRates rates[];
//--- Il contatore dei tentativi
   int attempts=0;
//--- Quanto è stato copiato
   int copied=0;
//--- Fa 25 tentativi per ottenere una timeseries sul simbolo desiderato
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- Se non è riuscito a copiare un numero sufficiente di barre
   if(copied!=bars)
     {
      //--- Forma un messaggio stringa
      string comm=StringFormat("Per il simbolo %s, è riuscito a ricevere solo %d barre
                               name,
                               copied,
                               bars
                               );
      //--- Mostra un messaggio in un commento nella finestra del grafico principale
      Comment(comm);
      //--- Mostra il messaggio
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Imposta il display del simbolo
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- Inizializza i buffers con valori vuoti
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
```

```
   ArrayInitialize(BarsBuffer4,0.0);
//--- Copia prezzi nei buffers
   for(int i=0;i<copied;i++)
     {
      //--- Calcola l'indice appropriato per i buffers
      int buffer_index=total-copied+i;
      //--- Scrive i prezzi nei buffers
      BarsBuffer1[buffer_index]=rates[i].open;
      BarsBuffer2[buffer_index]=rates[i].high;
      BarsBuffer3[buffer_index]=rates[i].low;
      BarsBuffer4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Restituisce in modo casuale il simoblo dal Market Watch                      |
//+------------------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Il numero dei simboli mostrati nella finestra Market Watch
   int symbols=SymbolsTotal(true);
//--- La posizione di un simbolo nella lista - un numero casuale da 0 a simboli
   int number=MathRand()%symbols;
//--- Restituisce il nome di un simbolo nella posizione specificata
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------------------+
//| Cambia l'apparenza delle barre                                               |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione delle informazioni riguardo le proprietà della ba
   string comm="";
//--- Un blocco per il cambio di colore delle barre
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore della linea
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Un blocco per il cambio della larghezza delle barre
   number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La larghezza è impostata da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
```

```
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Scrive il nome del simbolo
   comm="\r\n"+symbol+comm;

//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_CANDLES

Lo stile DRAW_CANDLES disegna candele sui valori di quattro buffer di indicatori, che contengono i prezzi Open, High, Low and Close prices. E' utilizzato per la creazione di indicatori personalizzati come una sequenza di candele, comprese quelle in una sottofinestra separata di un grafico e su altri strumenti finanziari.

Il colore delle candele può essere impostato tramite le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambia a seconda della situazione attuale.

L'indicatore è disegnato solo per queste barre, per cui i valori non vuoti di **tutti** e quattro i buffer indicatore sono impostati. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nelle proprietà PLOT_EMPTY_VALUE:

```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

Riempie esplicitamente i valori dei buffer indicatori, imposta un valore vuoto in un buffer per saltare le barre.

Il numero di buffer richiesti per il plotting di DRAW_CANDLES è 4. Tutti i buffer per il plotting dovrebbero andare uno dopo l'altro, nell'ordine: Open, High, Low e Close. Nessuno dei buffer può contenere solo valori vuoti, perché in questo caso nulla è tracciato.

È possibile impostare fino a tre colori per lo stile DRAW_CANDLES che influisce sull'aspetto della candela. Se un solo colore è impostato, viene applicata a tutte le candele sul chart.

```
//--- candele identiche con un unico colore applicato ad esse
#property indicator_label1  "Candele a colore unico"
#property indicator_type1   DRAW_CANDLES
//--- viene specificato un solo colore, quindi tutte le candele sono dello stesso colo
#property indicator_color1  clrGreen
```

Se vengono specificati due colori separati da virgola, il primo viene applicato al bordo della candela, mentre il secondo è applicato al corpo.

```
//--- Colori diversi per candele e stoppini
#property indicator_label1  "Candele a due colori"
#property indicator_type1   DRAW_CANDLES
//--- il verde viene applicato stoppini e contorni, mentre il bianco è applicato al co
#property indicator_color1  clrGreen,clrWhite
```

Specificare tre colori separati da virgola in modo che le candele che salgono e scendono vengono visualizzate in modo diverso. In tal caso, il primo colore viene applicato ai contorni candela, mentre il secondo e il terzo - a candele rialzista e ribassista.

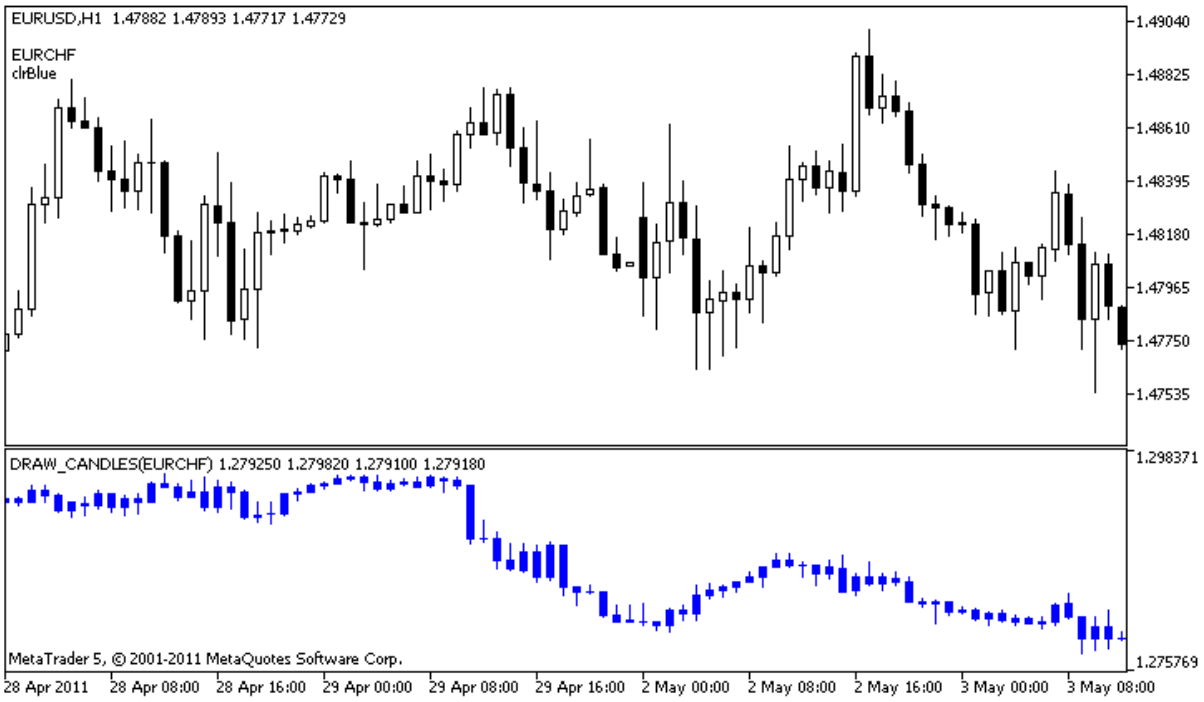
```
//--- Colori diversi per candele e stoppini
#property indicator_label1  "Candele a colore unico"
#property indicator_type1   DRAW_CANDLES
//--- stoppini e contorni sono di colore verde, il corpo della candela rialzista è bia
#property indicator_color1  clrGreen,clrWhite,clrRed
```

Quindi, lo stile DRAW_CANDLES permette di creare opzioni di colorazione personalizzata della candela. Inoltre, tutti i colori possono essere modificati dinamicamente durante il funzionamento dell'indicatore utilizzando la funzione PlotIndexSetInteger (composition_index_DRAW_CANDLES, PLOT_LINE_COLOR, modifier_index, color), dove modifier_index può avere i seguenti valori:

- 0 - colori di contorni e stoppini
- 1- colore del corpo della candela rialzista
- 2 - colore del corpo della candela ribassista

```
// --- impostare il colore dei contorni e stoppini
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
// --- impostare il colore del corpo della rialzista
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
// --- imposta il colore del corpo della ribassista
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

Un esempio di indicatore che disegna candele per uno strumento finanziario selezionato in una finestra separata. Il colore delle candele cambia dinamicamente ogni **N** ticks. Il parametro N è impostato nei parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà dell' indicatore).



Si prega di notare che per **plot1**, il colore è impostato usando la direttiva del compilatore #property, e poi nella funzione OnCalculate() il colore è impostato in modo casuale da una lista precedentemente preparata.

```
//+------------------------------------------------------------------+
//|                                                DRAW_CANDLES.mq5  |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per mostrare DRAW_CANDLES."
#property description "Disegna le candele di un simbolo selezionato in una finestra se
#property description " "
#property description "Il colore e lo spessore delle candele, così come il simbolo ver
#property description "dinamicamente ogni N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parametri di input
input int      N=5;              // Il numero di ticks per cambiare il tipo
input int      bars=500;         // Il numero di barre da mostrare
input bool     messages=false;   // Mostra i messaggi nel log "Expert Advisors"
//--- Buffer Indicatore
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- Nome simbolo
string symbol;
//--- Un array per salvare i colori
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
/ / --- Se bars è molto piccolo - completa il lavoro prima del tempo
   if(bars<50)
     {
      Comment("Prego specificare un gran numero di barre! L'operazione dell'indicatore
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mappatura buffers indicatore
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- Un valore vuoto
```

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Il nome del simbolo, per cui le barre vengono disegnate
   symbol=_Symbol;
//--- Imposta il display del simbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero sufficiente di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Imposta un nuovo simbolo dalla finestra Market Watch
      symbol=GetRandomSymbolName();
      //--- Cambia la forma
      ChangeLineAppearance();
      //--- Imposta un nuovo simbolo dalla finestra Market Watch
      int tries=0;
      //--- Fa 5 tentativi per riempire i buffers di plot1 con i prezzi dal simbolo
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- Un contatore delle chiamate della funzione CopyFromSymbolToBuffers()
         tries++;
        }
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

```
//+------------------------------------------------------------------------------------+
//| Riempie le candele specificate                                                     |
//+------------------------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                             )
  {
//--- Nell'array rates[], copieremo Open, High, Low e Close
   MqlRates rates[];
//--- Il contatore dei tentativi
   int attempts=0;
//--- Quanto è stato copiato
   int copied=0;
//--- Fa 25 tentativi per ottenere una timeseries sul simbolo desiderato
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- Se non è riuscito a copiare un numero sufficiente di barre
   if(copied!=bars)
     {
      //--- Forma un messaggio stringa
      string comm=StringFormat("Per il simbolo %s, è riuscito a ricevere solo %d barre
                               name,
                               copied,
                               bars
                               );
      //--- Mostra un messaggio in un commento nella finestra del grafico principale
      Comment(comm);
      //--- Mostra il messaggio
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Imposta il display del simbolo
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
     }
//--- Inizializza i buffers con valori vuoti
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
```

```
   ArrayInitialize(buff4,0.0);
//--- Su ogni tick, copia i prezzi dei buffers
   for(int i=0;i<copied;i++)
     {
      //--- Calcola l'indice appropriato per i buffers
      int buffer_index=total-copied+i;
      //--- Scrive i prezzi nei buffers
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Restituisce in modo casuale il simoblo dal Market Watch                      |
//+------------------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Il numero dei simboli mostrati nella finestra Market Watch
   int symbols=SymbolsTotal(true);
//--- La posizione di un simbolo nella lista - un numero casuale da 0 a simboli
   int number=MathRand()%symbols;
//--- Restituisce il nome di un simbolo nella posizione specificata
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------------------+
//| Cambia l'apparenza delle barre                                               |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione delle informazioni riguardo le proprietà della ba
   string comm="";
//--- Un blocco per il cambio di colore delle barre
   int number=MathRand(); // Ottiene un numero casuale
//--- Il divisore è uguale alla grandezza dell'array colors []
   int size=ArraySize(colors);
//--- Ottiene l'indice per selezionare un nuovo colore, come il resto della divisione
   int color_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Scrive il colore
   comm=comm+"\r\n"+(string)colors[color_index];
//--- Scrive il nome del simbolo
   comm="\r\n"+symbol+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_LINE

Il valore DRAW_COLOR_LINE è una variante colorata dello stile DRAW_LINE e disegna una linea con i valori del buffer di indicatore. Ma questo stile, come tutti gli stili di colore con la parola **COLOR** nel loro titolo ha un buffer indicatore aggiuntivo speciale che memorizza l'indice di colore (numero) da un array di di colori impostato in modo speciale. Così, il colore di ogni segmento della linea può essere definito specificando l'indice di colore dell'indice per disegnare la linea in questa barra.

La larghezza, stile e colori delle linee possono essere impostati utilizzando le direttive del compilatore e dinamicamente utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambia a seconda della situazione attuale.

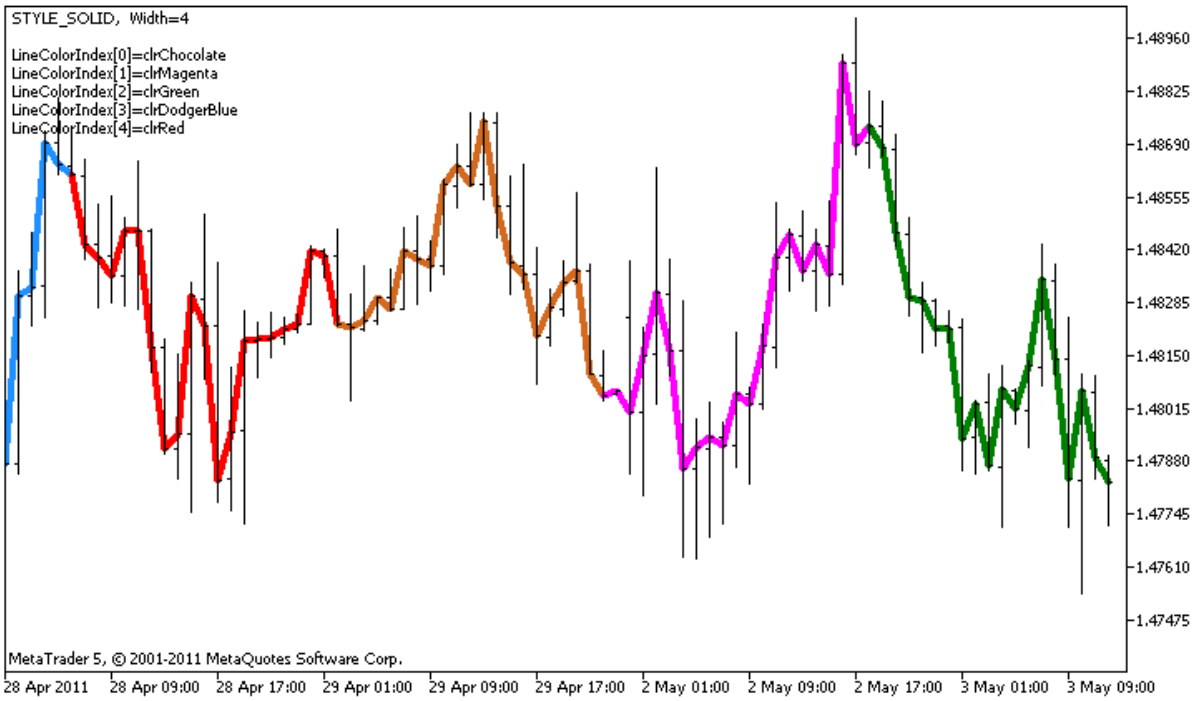
Il numero di buffer necessari per tracciare DRAW_COLOR_LINE è 2.

- un buffer per memorizzare i valori degli indicatori utilizzati per il disegno di una linea;
- un buffer per memorizzare l'indice del colore della linea su ciascuna barra.

I colori possono essere specificati con la direttiva del compilatore #property indicator_color1 separati da una virgola. Il numero di colori non può superare 64.

```
//--- Definire 5 colori per colorare ogni barra (sono memorizzati nell' array speciale
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (Up to 64
```

Un esempio di indicatore che disegna una linea utilizzando i prezzi Close delle barre. La larghezza della linea e lo stile cambiano in modo casuale ogni N=5 ticks.



I colori dei segmenti di linea anche cambiano anche in modo casuale nella funzione personalizzata ChangeColors().

```
//+------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti della linea                                    |
```

```
//+------------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione intera
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scriviar
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

L'esempio mostra la caratteristica delle versioni "colore" degli indicatori - per modificare il colore di un segmento di una linea, non è necessario modificare i valori nel buffer ColorLineColors[] (che contiene gli indici del colore). Tutto quello che dovete fare è impostare nuovi colori in un'array speciale. Questo permette di cambiare rapidamente il colore una volta per l'intero plotting, cambiando solo una piccola gamma di colori utilizzando la funzione PlotIndexSetInteger().

Notare che inizialmente per **Plot1** con DRAW_COLOR_LINE le proprietà vengono impostate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() queste tre proprietà sono impostate in modo casuale.

I parametri N e lunghezza (la lunghezza dei segmenti di colore nei bar) i parametri sono impostati nei parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                      DRAW_COLOR_LINE.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "Un indicatore che dimostra DRAW_COLOR_LINE"
#property description "Traccia una linea sul prezzo di chiusura in pezzi colorati di 2
#property description "La larghezza, lo stile e il colore delle parti della linea venq
#property description "ogni N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- Definire 5 colori per colorare ogni barra (sono memorizzati nell' array speciale
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (Up to 64
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      N=5;           //Numero di ticks da cambiare
input int      Length=20;     // La lunghezza di ciascuna parte di colore in barre
int            line_colors=5; // Il numero di colori impostati è 5 - vedi #property in
//--- Un buffer per il plotting
double         ColorLineBuffer[];
//--- Un buffer per memorizzare il colore linea su ciascuna barra
double         ColorLineColors[];

//--- L'array per memorizzare i colori contiene 7 elementi
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- Associa un array e un buffer indicatore
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- Inizializzazione del generatore di numeri pseudo-casuali
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Cambia i colori delle sezioni della linea
      ChangeColors(colors,5);
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Blocco per il calcolo dei valori dell'indicatore
   for(int i=0;i<rates_total;i++)
     {
      //--- Scrive il valore dell'indicatore nel buffer
      ColorLineBuffer[i]=close[i];
      //--- Ora, imposta casualmente un set di indici di colori per questa barra
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- Per questa barra, la linea avrà il colore con l'indice color_index
      ColorLineColors[i]=color_index;
     }

//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti della linea                                    |
//+------------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
```

```
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione intera
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scrivian
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Cambia l'apparenza di una linea visualizzata nell'indicatore                 |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco per modificare la larghezza della linea
   int number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5; // Lo spessore è impostato da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+" Width="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   int size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_SECTION

The DRAW_COLOR_SECTION style is a color version of DRAW_SECTION, but unlike the latter, it allows drawing sections of different colors. Lo stile DRAW_COLOR_SECTION, come tutti gli stili di colore con la parola **COLOR** nel titolo, contiene un buffer indicatore aggiuntivo speciale che memorizza l'indice del colore (numero) da un array di di colori impostato in modo speciale. Così, il colore di ogni sezione può essere definito specificando l'indice di colore dell'indice della barra che corrisponde alla fine sezione.

The width, color and style of the sections can be specified like for the DRAW_SECTION style - using compiler directives or dynamically using the PlotIndexSetInteger() function. Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambi a seconda della situazione attuale.

Le sezioni vengono disegnate da un valore non vuoto di un altro valore non vuoto del buffer indicatore, i valori vuoti vengono ignorati. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nella proprietà PLOT_EMPTY_VALUE: ad esempio, se l'indicatore deve essere disegnato come una sequenza di sezioni su valori diversi da zero, allora è necessario impostare il valore zero come uno vuoto:

```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

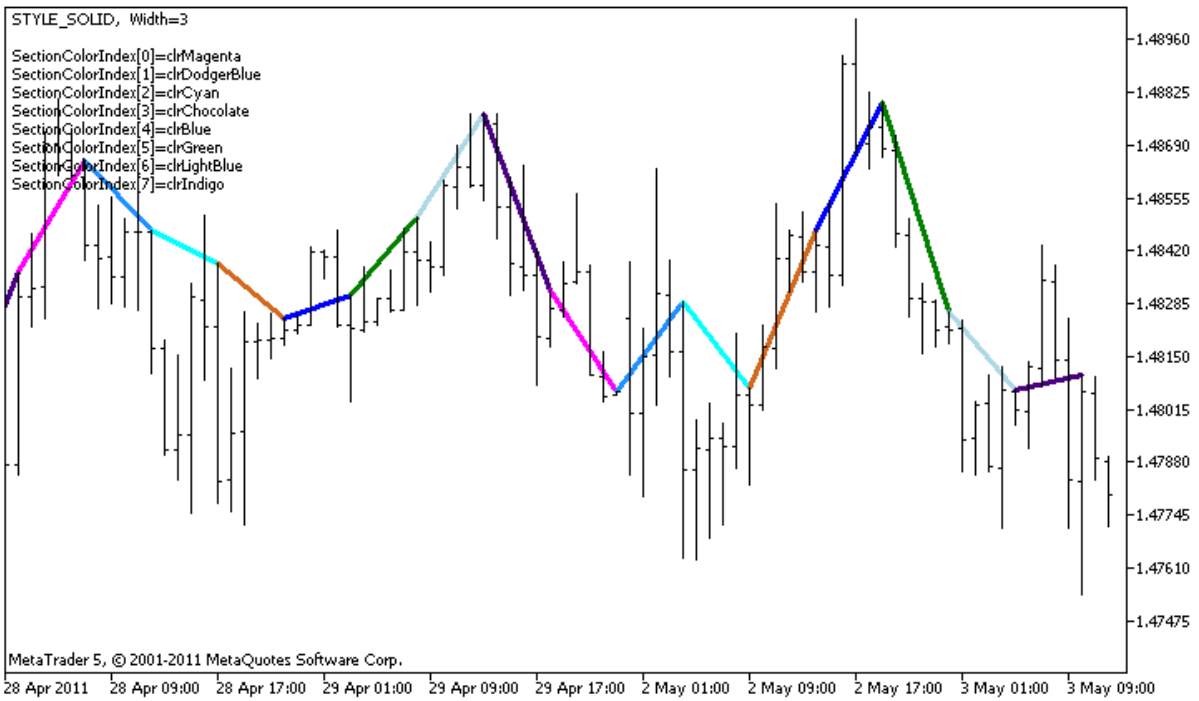
Riempire sempre esplicitamente i valori dei buffer indicatore, impostare un valore vuoto in un buffer per gli elementi che non devono essere tracciati.

Il numero di buffer necessari per tracciare DRAW_COLOR_SECTION è 2.

- un buffer per memorizzare i valori degli indicatori utilizzati per il disegno di una linea;
- un buffer per memorizzare l'indice del colore, che viene utilizzato per elaborare la sezione (ha senso impostare solo valori non vuoti).

I colori possono essere specificati con la direttiva del compilatore #property indicator_color1 separati da una virgola. Il numero di colori non può superare 64.

Un esempio di indicatore che trae sezioni colorate ciascuna lunga 5 barre, utilizzando i valori del prezzo High. Il colore, la larghezza e lo stile delle sezioni cambiano in modo casuale ogni **N** ticks.



Notare che inizialmente per **Plot1** con DRAW_COLOR_SECTION 8 colori vengono impostati utilizzando la direttiva del compilatore #property. Poi nella funzioneOnCalculate(), i colori vengono impostati in modo casuale dalla gamma di colori colori[].

Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                    DRAW_COLOR_SECTION.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                    https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per mostrare DRAW_COLOR_SECTION"
#property description "Esso disegna sezioni colorate con la lunghezza uguale al numero
#property description "Il colore, spessore e stile delle sezioni vengono cambiati in r
#property description "dopo ogni N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorSection
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- Definisce 8 colori per la colorazione delle sezioni selezione (vengono conservat
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,c
#property indicator_style1  STYLE_SOLID
```

```
#property indicator_width1  1
//--- parametri di input
input int      N=5;                      // Numero di ticks da cambiare
input int      bars_in_section=5;        // La lunghezza delle sezioni espressa in bar
//--- Una variabile ausiliaria per calcolare la fine delle sezioni
int            divider;
int            color_sections;
//--- Un buffer per il plotting
double         ColorSectionBuffer[];
//--- Un buffer per memorizzare il colore linea su ciascuna barra
double         ColorSectionColors[];
//--- Un array per memorizzare i colori contiene 14 elementi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- Il numero di colori ai colori delle sezioni
   int color_sections=8;   //  mostra un commento a #property indicator_color1
//--- Controllare il parametro indicatore
   if(bars_in_section<=0)
     {
      PrintFormat("Invalid section length=%d",bars_in_section);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Cambia colori usati per tracciare le sezioni
      ChangeColors(colors,color_sections);
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Il numero della barra da cui inizia il calcolo dei valori degli indicatori
   int start=0;
//--- Se l'indicatore è stato calcolato prima, allora impostare l'inizio sulla barra p
   if(prev_calculated>0) start=prev_calculated-1;
//--- Qui ci sono tutti i calcoli dei valori degli indicatori
   for(int i=start;i<rates_total;i++)
     {
      //--- Se il numero della barra è divisibile per il section_length, significa che
      if(i%bars_in_section==0)
        {
         //--- Imposta la fine della sezione presso il prezzo elevato di questo bar
         ColorSectionBuffer[i]=high[i];
         //--- Un resto della divisione del numero di barra per number_of_colors*sceti
         int rest=i%divider;
         //Ottiene il numero dei colori =  da 0 a number_of_colors-1
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //--- Se il resto della divisione è pari a barre,
      else
        {
         //--- Se non è successo niente, ignora bar - set 0
         else ColorSectionBuffer[i]=0;
        }
     }
//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
```

```
//| Cambia il colore dei segmenti della linea                                |
//+--------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione intera
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scrivia
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToSt
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+--------------------------------------------------------------------------+
//| Cambia l'apparenza di una linea visualizzata nell'indicatore             |
//+--------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco per modificare la larghezza della linea
   int number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5; // Lo spessore è impostato da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+" Width="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   int size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
```

```
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_HISTOGRAM

Lo stile DRAW_COLOR_HISTOGRAM disegna un istogramma come una sequenza di colonne colorate da zero ad un valore specificato. I valori sono presi dal buffer indicatore. Ogni colonna può avere il proprio colore da un insieme predefinito di colori.

La larghezza, il colore e lo stile dell'istogramma possono essere specificati come per lo stile DRAW_HISTOGRAM - utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di cambiare l'aspetto dell'istogramma sulla base della situazione attuale.

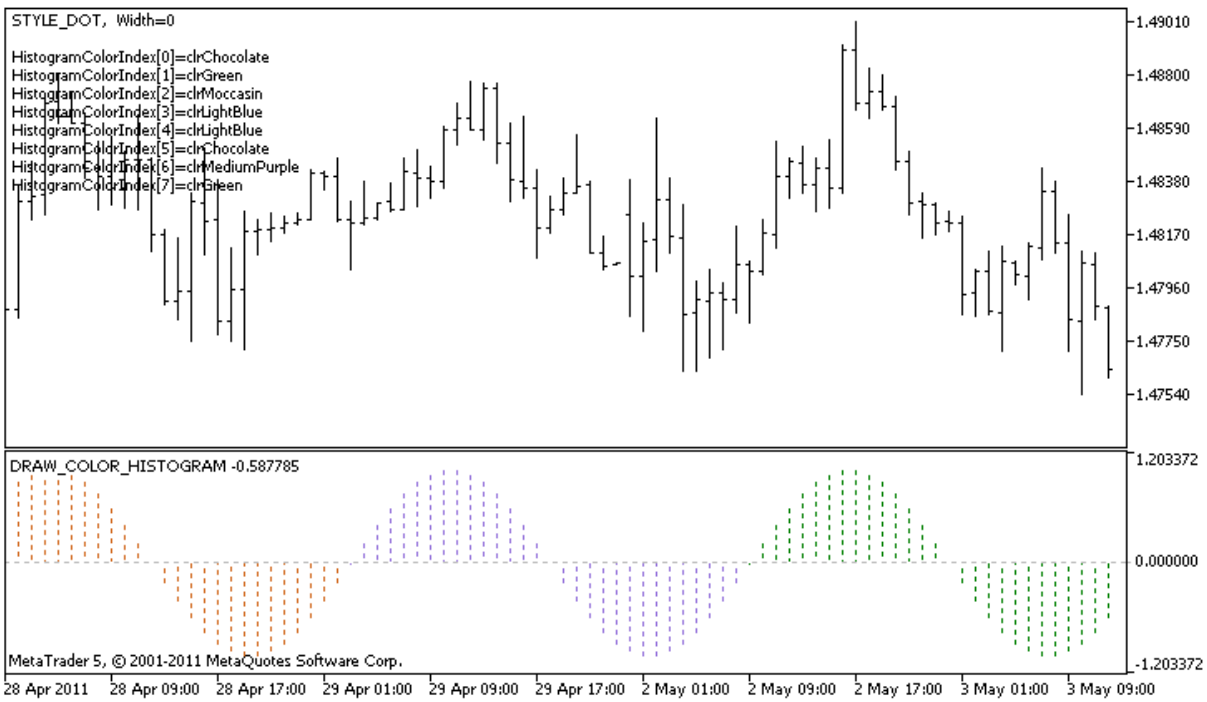
Dal momento che una colonna dal livello zero è disegnata su ogni barra, DRAW_COLOR_HISTOGRAM dovrebbe meglio essere utilizzata in una finestra di grafico separata. Molto spesso questo tipo di plotting viene utilizzato per creare indicatori del tipo oscillatore, per esempio, Awesome Oscillator oppure Market Facilitation Index. Per i valori vuoti non visualizzabili deve essere specificato il valore zero.

Il numero di buffer necessari per tracciare DRAW_COLOR_HISTOGRAM è 2.

- un buffer per memorizzare un valore non-zero del segmento verticale su ciascuna barra, la seconda estremità del segmento è sempre sulla linea zero dell'indicatore;
- un buffer per memorizzare l'indice del colore, che viene utilizzato per elaborare la sezione (ha senso impostare solo valori non vuoti).

I colori possono essere specificati utilizzando la direttiva del compilatore #property indicator_color1 separata da una virgola. Il numero di colori non può superare 64.

Un esempio di indicatore che disegna una sinusoide di un colore specificato basato sulla funzione MathSin(). Il colore, la larghezza e lo stile di **tutte** le colonne dell'istogramma cambiano in modo casuale ogni **N** ticks. Il parametro barre specifica il periodo della sinusoide, che è dopo il numero specificato di barre per cui la sinusoide ripeterà il ciclo.



Si prega di notare che per **Plot1** con lo stile DRAW_COLOR_HISTOGRAM, 5 colori vengono impostati utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() i colori sono scelti a caso dai 14 colori memorizzati nell' array colors[] . Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                    DRAW_COLOR_HISTOGRAM.mq5 |
//|                                 Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per mostrare DRAW_COLOR_HISTOGRAM"
#property description "Esso disegna una sinusoide come un istogramma in una finestra s
#property description "Il colore e la larghezza delle colonne vengono modificate in mo
#property description "dopo ogni N ticks"
#property description "Il parametro barre imposta il numero di barre per ripetere la s

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- parametri di input
input int      bars=30;          // Il periodo della sinusoide in barre
input int      N=5;              // Il numero di ticks che cambiano l'istogramma
//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- Definisce 8 colori per la colorazione delle sezioni selezione (vengono conservat
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Un buffer di valori
double         Color_HistogramBuffer[];
//--- Un buffer di indici di colore
double         Color_HistogramColors[];
//--- Un fattore per ottenere l'angolo di 2Pi in radianti, moltiplicato per il paramet
double         multiplier;
int            color_sections;
//--- Un array per memorizzare i colori contiene 14 elementi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
```

```
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- Il numero di colori per colorare la sinusoide
   color_sections=8;   //  vedi un commento a #property indicator_color1
//--- Calcola il moltiplicatore
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Imposta il valore di bars=%d maggiore di 1",barre);
      //--- Terminazione precoce dell'indicatore
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Cambia i colori usati nell'istogramma
      ChangeColors(colors,color_sections);
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }
```

```
//--- Calcola i valori dell'indicatore
   int start=0;
//--- Se già calcolato nelle precedenti starts di OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // imposta l'inizio del calcolo con
//--- Compila il buffer di indicatore con valori
   for(int i=start;i<rates_total;i++)
     {
      //--- Un valore
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- Colore
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti della linea                                    |
//+------------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione intera
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scrivian
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Cambia l'apparenza di una linea visualizzata nell'indicatore                 |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
```

```
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco per modificare la larghezza della linea
   int number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5; // Lo spessore è impostato da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+" Width="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   int size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_HISTOGRAM2

Lo stile DRAW_COLOR_HISTOGRAM2 disegna un istogramma di un colore specificato - segmenti verticali utilizzando i valori di due buffer indicatore. Ma a differenza del mono-colore DRAW_HISTOGRAM2, in questo stile ogni colonna dell'istogramma può avere il suo colore da un insieme predefinito. I valori di tutti i segmenti vengono presi dal buffer indicatore.

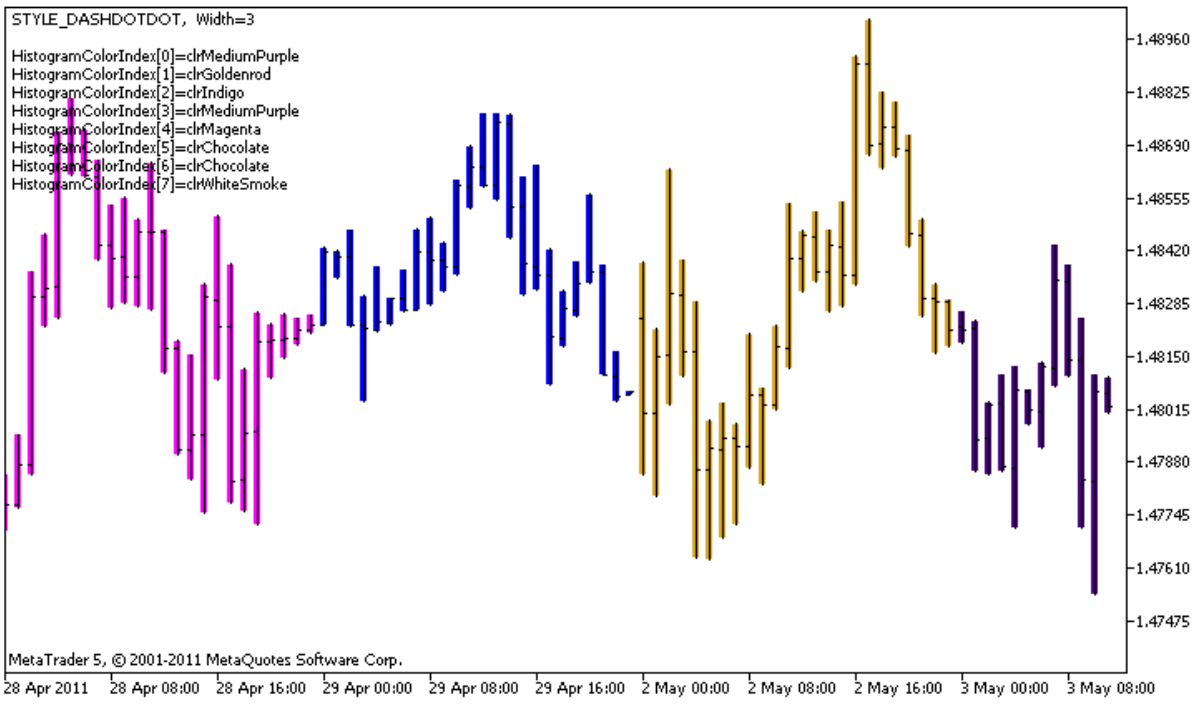
La larghezza, stile e colore dell'istogramma possono essere specificati come per lo stile DRAW_HISTOGRAM2 - utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di cambiare l'aspetto dell'istogramma sulla base della situazione attuale.

Lo stile DRAW_COLOR_HISTOGRAM2 può essere utilizzato in una sottofinestra separata di un grafico e nella sua finestra principale. Per i valori vuoti nulla viene tracciato, tutti i valori nei buffer indicatore devono essere impostati in modo esplicito. I buffer non vengono inizializzati con i valori vuoti.

Il numero di buffer necessari per tracciare DRAW_COLOR_HISTOGRAM2 è 3:

- due buffer per memorizzare l'estremità superiore e inferiore del segmento verticale su ciascuna barra;
- un buffer per memorizzare l'indice del colore, che viene utilizzato per disegnare il segmento (ha senso impostare solo valori non vuoti).

Un esempio di indicatore che disegna un istogramma di colore specificato tra i prezzi Alti e Bassi. Per ogni giorno della settimana, le linee dell'istogramma hanno un colore differente. Il colore del giorno, la larghezza e lo stile dell'istogramma cambiano in modo casuale ogni **N** ticks.



Si prega di notare che per **Plot1** con lo stile DRAW_COLOR_HISTOGRAM2, 5 colori vengono impostati utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() i colori sono scelti a caso dai 14 colori memorizzati nell' array colors[] .

Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------------+
//|                                                      DRAW_COLOR_HISTOGRAM2.mq5 |
//|                                    Copyright 2011, MetaQuotes Software Corp. |
//|                                                          https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_COLOR_HISTOGRAM2"
#property description "Disegna un segmento tra Open e Close su ciascuna barra"
#property description "Il colore, lo spessore e lo stile vengono cambiati casualmente"
#property description "dopo ogni N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot ColorHistogram_2
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- Definisce 5 colori per colorare l'istogramma in base ai giorni della settimana
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parametri input
input int      N=5;              // Il numero di ticks che cambiano l'istogramma
int            color_sections;
//--- Valori dei buffer
double         ColorHistogram_2Buffer1[];
double         ColorHistogram_2Buffer2[];
//--- Un buffer di indici di colore
double         ColorHistogram_2Colors[];
//--- Un array per memorizzare i colori contiene 14 elementi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                           |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
```

```
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//--- Imposta un valore vuoto
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- Il numero di colori per colorare la sinusoide
   color_sections=8;   //  Vedere il commento a #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Cambia i colori usati per disegnare l'istogramma
      ChangeColors(colors,color_sections);
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Calcola i valori dell'indicatore
   int start=0;
//--- Per ottenere il giorno della settimana per il prezzo di apertura di ogni barra
   MqlDateTime dt;
//--- Se già calcolato nelle precedenti starts di OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // imposta l'inizio del calcolo con
//--- Compila il buffer di indicatore con valori
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
```

```
      //--- valore
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
      //--- Imposta l'indice del colore secondo il giorno della settimana
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- Restituisce il valore prev_calculated per la successiva chiamata della funzione
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti della linea                                          |
//+------------------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione dell'intero
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scriviar
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------------------------+
//| Cambia l'apparenza di una linea visualizzata nell'indicatore                       |
//+------------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco per modificare la larghezza della linea
   int number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5; // Lo spessore è impostato da 0 a 4
```

```
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+" Width="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   int size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_ARROW

Lo stile DRAW_COLOR_ARROW disegna frecce colorate (simboli del set Wingdings) basati sui valori del buffer indicatore. In contrasto a DRAW_ARROW, in questo stile è possibile impostare un colore da un insieme predefinito di colori specificato dalle proprietà di indicator_color1 per ogni simbolo.

La larghezza ed il colore dei simboli possono essere specificati come per lo stile DRAW_ARROW - utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di cambiare l'aspetto di un indicatore sulla base della situazione attuale.

Il codice simbolo viene impostato utilizzando la proprietà PLOT_ARROW

```
//--- Definisce il codice simbolo dal carattere Wingdings per disegnare in PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Il valore predefinito di PLOT_ARROW=159 (un cerchio).

Ogni freccia è in realtà un simbolo che ha l'altezza ed il punto di ancoraggio, e può coprire alcune importanti informazioni su un grafico (per esempio, il prezzo di chiusura alla bar). Quindi, possiamo anche specificare lo slittamento verticale in pixel, che non dipende dalla scala del grafico. Le frecce saranno spostate verso il basso per il numero di pixel specificato, anche se i valori dell'indicatore rimarranno gli stessi:

```
//--- Imposta lo spostamento verticale di frecce in pixel
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

Un valore negativo di PLOT_ARROW_SHIFT significa lo spostamento delle frecce verso l'alto, un valore positivo slitta la freccia verso il basso.

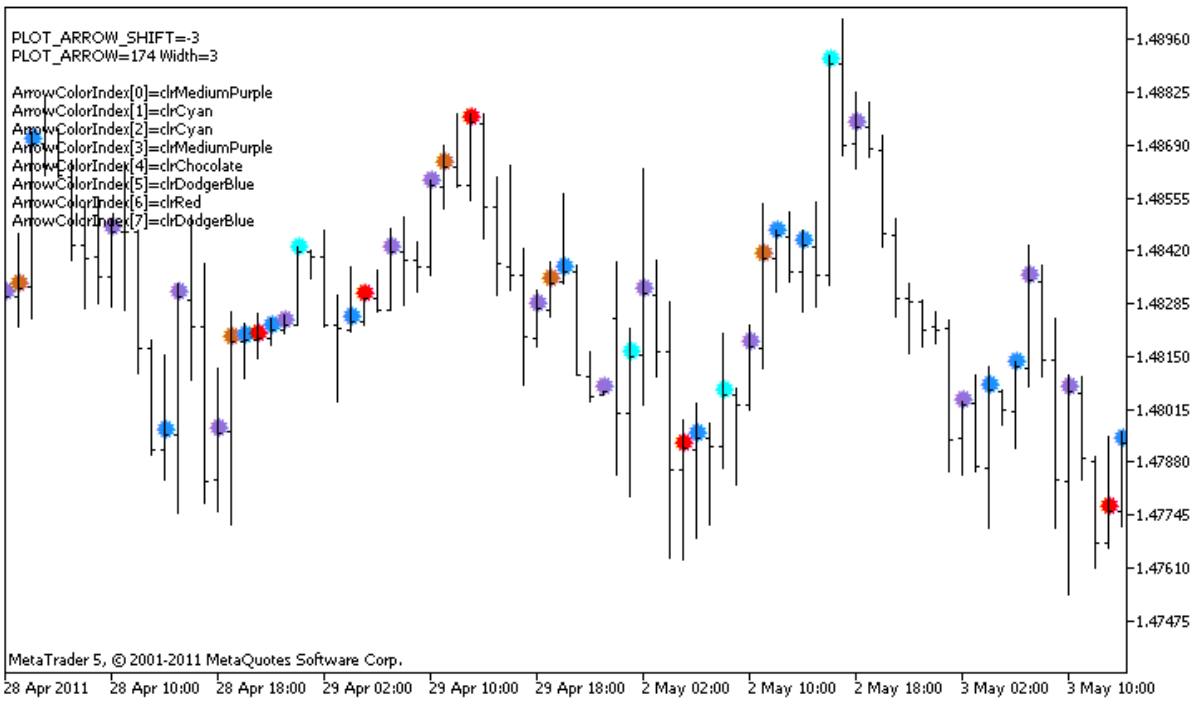
Lo stile DRAW_COLOR_ARROW può essere utilizzato in una sottofinestra separata di un grafico e nella sua finestra principale. Valori vuoti non vengono disegnati e non compaiono nella "Finestra Dati"; tutti i valori dei buffer indicatori devono essere impostati in modo esplicito. I buffer non vengono inizializzati con un valore zero.

```
//--- Imposta un valore vuoto
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);
```

Il numero di buffer necessari per il plotting di DRAW_COLOR_ARROW è 2.

- un buffer per memorizzare il valore del prezzo che viene utilizzato per disegnare il simbolo (più uno slittamento in pixel, dato nella proprietà PLOT_ARROW_SHIFT);
- un buffer per memorizzare l'indice di colore, che viene utilizzato per disegnare una freccia (ha senso impostare solo valori non vuoti).

Un esempio di indicatore, che disegna frecce su ciascuna barra con il prezzo close superiore al prezzo di chiusura della barra precedente. La larghezza, slittamento e codice simbolo di **tutte** le frecce vengono cambiate in modo casuale ogni **N** ticks. Il colore del simbolo dipende dal numero della barra su cui è disegnato.



Nell'esempio, per **plot1** con lo stile DRAW_COLOR_ARROW, la proprietà, il colore e le dimensioni vengono specificate utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() le proprietà sono impostate in modo casuale. Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

Si prega di notare che inizialmente vengono impostati 8 colori utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate(), il colore viene impostato in modo casuale da 14 colori che sono memorizzati nella matrice colors[].

```
//+------------------------------------------------------------------------------+
//|                                                      DRAW_COLOR_ARROW.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per mostrare DRAW_COLOR_ARROW"
#property description "Disegna varie frecce-colore impostate da caratteri Unicode, su
#property description "Il colore, grandezza, slittamento e codice del simbolo delle fi
#property description "dinamicamente ogni N ticks"
#property description "Il parametro code imposta il valore base: code=159 (un cerchio)

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorArrow
#property indicator_label1  "ColorArrow"
```

```
#property indicator_type1   DRAW_COLOR_ARROW
//--- Definisce 8 colori per colorare l'istogramma in base ai giorni della settimana
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parametri di input
input int       N=5;          // Numero di ticks da cambiare
input ushort    code=159;     // Codice simbolo da disegnare in DRAW_ARROW
int             color_sections;
//--- Un buffer indicatore per il disegno
double          ColorArrowBuffer[];
//--- Un buffer per memorizzare gli indici di colore
double          ColorArrowColors[];
//--- Un array per memorizzare i colori contiene 14 elementi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- Definisce il codice simbolo per il disegnamento in PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- Imposta lo spostamento verticale di frecce in pixel
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- Imposta come un valore vuoto 0
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- Il numero di colori per colorare la sinusoide
   color_sections=8;   // vedere il commento di #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare il colore, la grandezza , slittamento e codice del
   ticks++;
//--- Se un numero critico di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà delle frecce
      ChangeLineAppearance();
      //--- Cambia i colori usati per disegnare l'istogramma
      ChangeColors(colors,color_sections);
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- Blocco per il calcolo dei valori dell'indicatore
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- Ciclo del calcolo
   for(int i=1;i<rates_total;i++)
     {
      //--- Se il prezzo Close corrente è maggiore di quello precedente, disegna una
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- In caso contrario, specificare il valore null
      else
         ColorArrowBuffer[i]=0;
      //--- Colore freccia
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti della linea                                    |
//+------------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
```

```
      {
       //--- Ottiene un valore casuale
       int number=MathRand();
       //--- Ottiene un indice nell'array col[] come resto della divisione dell'intero
       int i=number%size;
       //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
       PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                           PLOT_LINE_COLOR,      //  Identificatore proprietà
                           plot_color_ind,       //  Liindice del colore, dove scrivia
                           cols[i]);             //  Un nuovo colore
       //--- Scrivi i colori
       comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
       ChartSetString(0,CHART_COMMENT,comm);
      }
//---
  }
//+------------------------------------------------------------------------------------+
//| Cambia l'apparenza di una linea visualizzata nell'indicatore                       |
//+------------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di linea
   string comm="";
//--- Un blocco per modificare la larghezza della linea
   int number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5; // Lo spessore è impostato da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+" Width="+IntegerToString(width);

//--- Un blocco per il cambio del codice della freccia (PLOT_ARROW)
   number=MathRand();
//--- Ottiene il resto della divisione intera per calcolare un nuovo codice della fre
   int code_add=number%20;
//--- Imposta il nuovo codice simbolo come il risultato di code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- Scrive il codice simbolo PLOT_ARROW
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- Un blocco per cambiare lo slittamento verticale delle frecce in pixel
   number=MathRand();
//--- Prende lo slittamento come il resto della divisione intera
   int shift=20-number%41;
//--- Imposta il nuovo slittamento da
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- Scrive lo slittamento di PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
```

```
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_ZIGZAG

Lo stile DRAW_COLOR_ZIGZAG disegna segmenti di colori diversi, utilizzando i valori di due buffer indicatori. Questo stile è una versione colorata di DRAW_ZIGZAG, cioè permette di specificare per ogni segmento un singolo colore dal set predefinito di colori. I segmenti vengono tracciati da un valore nel primo buffer di un valore nel secondo buffer indicatore. Nessuno dei buffer può contenere solo valori vuoti, perché in questo caso nulla è tracciato.

La larghezza, il colore e lo stile della linea possono essere specificati come per lo stile DRAW_ZIGZAG - utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambi a seconda della situazione attuale.

Le sezioni vengono disegnate da un valore non vuoto di un buffer ad un valore non vuoto di un altro buffer indicatore. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nella proprietà PLOT_EMPTY_VALUE:

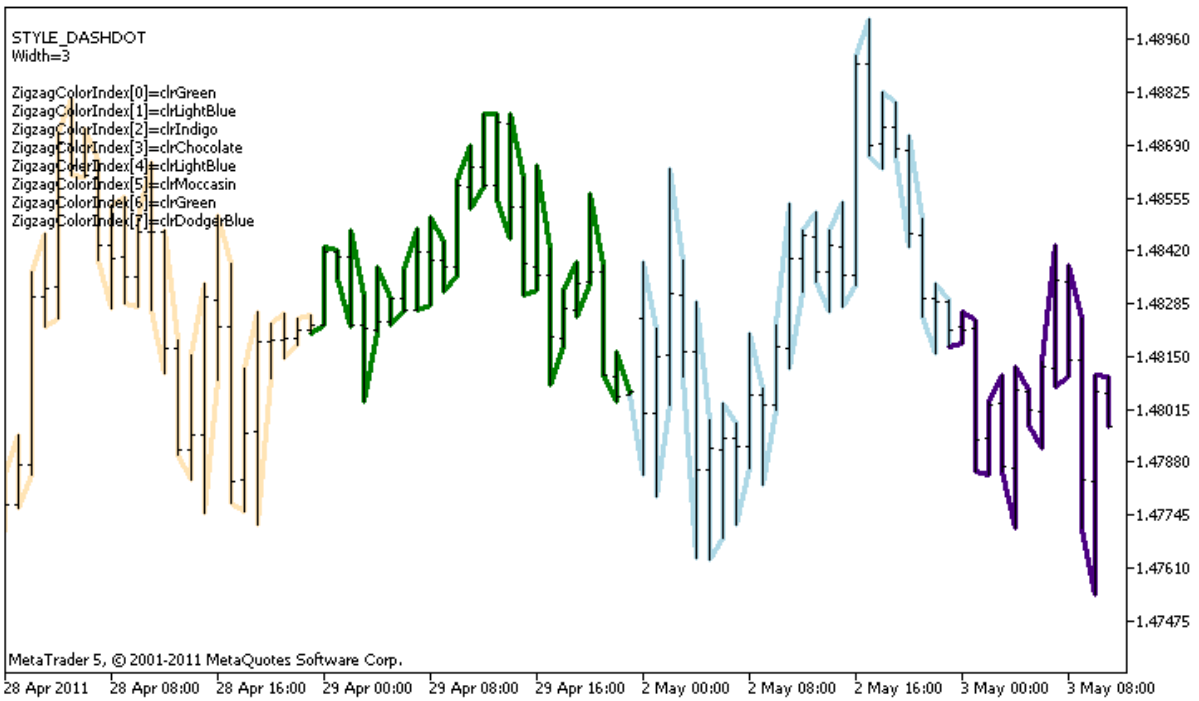
```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Sempre riempire esplicitamente i valori dei buffer indicatori, impostando un valore vuoto in un buffer per saltare le sbarre.

Il numero di buffer necessari per tracciare DRAW_COLOR_ZIGZAG è 3:

- due buffer per memorizzare i valori delle estremità delle sezioni zigzag;
- un buffer per memorizzare l'indice del colore, che viene utilizzato per elaborare la sezione (ha senso impostare solo valori non vuoti).

Un esempio di indicatore che traccia una sega sulla base dei prezzi High e Low. Il colore, la larghezza e lo stile delle linee a zig-zag cambiano in modo casuale ogni N ticks.



Si prega di notare che per **Plot1** con lo stile DRAW_COLOR_ZIGZAG, 8 colori vengono impostati utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() il colore è scelto a caso dai 14 colori memorizzati nell'array colors[].

Il parametro N è impostato in parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà del indicatore).

```
//+------------------------------------------------------------------------------+
//|                                                      DRAW_COLOR_ZIGZAG.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_COLOR_ZIGZAG"
#property description "Traccia una linea spezzata come una sequenza di sezioni colorat
#property description "Il colore, spessore e stile dei segmenti vengono cambiati in m
#property description " ogni N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot Color_Zigzag
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- Definisce 8 colori per la colorazione delle sezioni selezione (vengono conservat
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri input
input int      N=5;              // Numero di ticks da cambiare
int            color_sections;
//--- Buffer dei valori delle fini del segmento
double         Color_ZigzagBuffer1[];
double         Color_ZigzagBuffer2[];
//--- Buffer degli indici dei colori delle fini del segmento
double         Color_ZigzagColors[];
//--- Un array per memorizzare i colori contiene 14 elementi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Un array per memorizzare gli stili di linea
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
```

```
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//---- Numero di colori per colorare il zigzag
   color_sections=8;   //  mostra un commento in #property indicator_color1 property
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcola i ticks per cambiare lo stile, il colore e lo spessore della linea
   ticks++;
//--- Se un numero sufficiente di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Cambia colori usati per tracciare le sezioni
      ChangeColors(colors,color_sections);
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }

//--- La struttura del tempo è richiesta per ottenere il giorno della settimana di ogr
   MqlDateTime dt;

//--- La posizione di partenza dei calcoli
   int start=0;
//--- Se l'indicatore è stato calcolato sulla tick precedente, allora avviare il calc
   if(prev_calculated!=0) start=prev_calculated-1;
//--- Ciclo del calcolo
   for(int i=start;i<rates_total;i++)
```

```
     {
      //--- Scrive il tempo di apertura della barra nella struttura
      TimeToStruct(time[i],dt);

      //--- Se il numero è pari
      if(i%2==0)
        {
         //--- Scrive High nel 1° buffer e Low nel 2°
         Color_ZigzagBuffer1[i]=high[i];
         Color_ZigzagBuffer2[i]=low[i];
         //--- Il colore dei segmenti
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
      //--- il numero della barra è dispari
      else
        {
         //--- Riempie la barra in ordine inverso
         Color_ZigzagBuffer1[i]=low[i];
         Color_ZigzagBuffer2[i]=high[i];
         //--- Il colore dei segmenti
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti zigzag                                               |
//+------------------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione dell'intero
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scrivia
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
```

```
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Modifica l'aspetto dei segmenti a zig-zag                                    |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà di Color_ZigZag
   string comm="";
//--- Un blocco per modificare la larghezza della linea
   int number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La larghezza è impostata da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Un blocco per cambiare lo stile della linea
   number=MathRand();
//--- Il divisore è uguale alla grandezza dell'array styles
   int size=ArraySize(styles);
//--- Ottiene l'indice per selezionare un nuovo stile, come resto della divisione inte
   int style_index=number%size;
//--- Imposta il colore come la proprietà PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Scrive lo stile della linea
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_BARS

Lo stile DRAW_COLOR_BARS disegna barre sui valori delle quattro buffer di indicatori, che contengono i prezzi Open, High, Low e Close. Questo stile è una versione avanzata di DRAW_BARS e permette di specificare per ogni barra un singolo colore dal set predefinito di colori. E' utilizzato per la creazione di indicatori personalizzati come barre, incluse quelli in una sottofinestra separata di un grafico e su altri strumenti finanziari.

Il colore delle barre può essere impostato utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambia a seconda della situazione attuale.

L'indicatore è disegnato solo per queste barre, per cui i valori non vuoti di **tutti** e quattro i buffer indicatore sono impostati. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nelle proprietà PLOT_EMPTY_VALUE:

```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

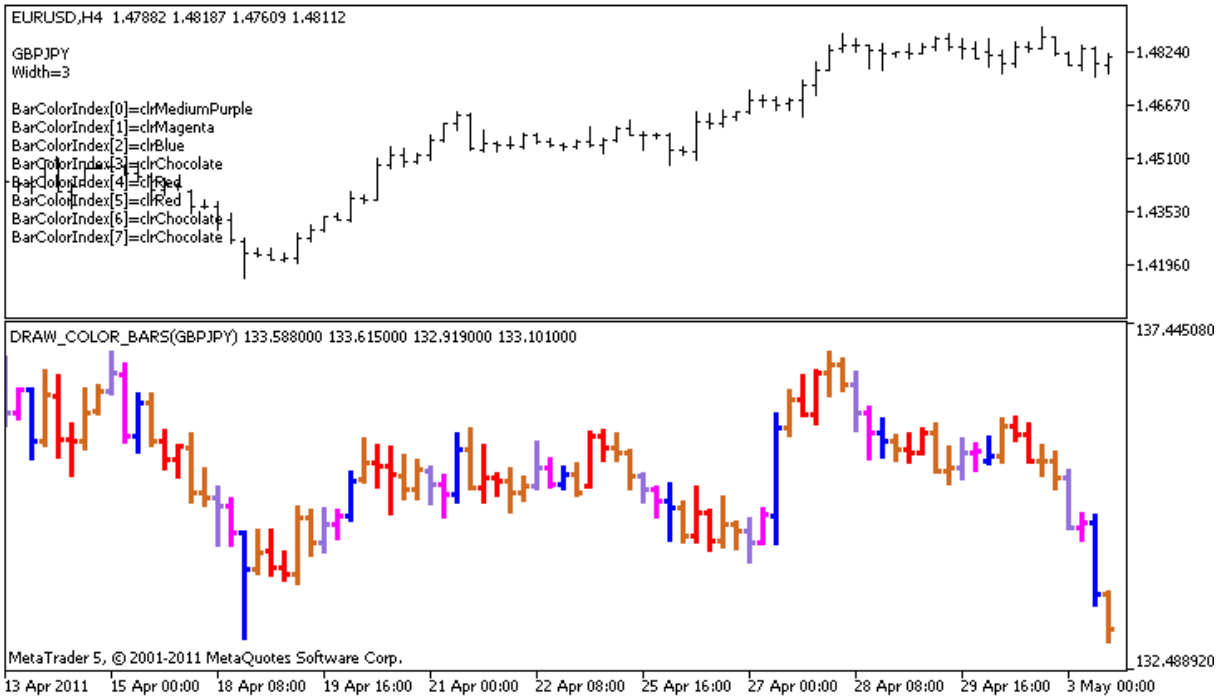
Riempie esplicitamente i valori dei buffer indicatori, imposta un valore vuoto in un buffer per saltare le barre.

Il numero di buffer richiesti per il plotting di DRAW_COLOR_BARS è 5:

- quattro buffer per memorizzare i valori di Open, High, Low e Close;
- un buffer per memorizzare l'indice del colore, che viene utilizzato per disegnare una barra (ha senso impostarlo solo per le barre che verranno disegnate).

Tutti i buffer per il tracciamento dovrebbero andare uno dopo l'altro, nell'ordine: Open, High, Low, Close e il buffer del colore. Nessuno dei buffer di prezzo possono contenere solo valori nulli, poiché in questo caso nulla è tracciato.

Un esempio di indicatore che disegna barre su uno strumento finanziario selezionato in una finestra separata. Il colore delle barre varia casualmente ogni **N** ticks. Il parametro N è impostato nei parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà dell' indicatore).



Si prega di notare che per **Plot1** con lo stile DRAW_COLOR_BARS, 8 colori vengono impostati utilizzando la direttiva del compilatore #property, e poi nella funzione OnCalculate() il colore è scelto a caso dai 14 colori memorizzati nell'array colori[].

```
//+------------------------------------------------------------------------------------+
//|                                                              DRAW_COLOR_BARS.mq5 |
//|                                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                              https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore che dimostra DRAW_COLOR_BARS"
#property description "Esso disegna barre di diverso colore di un simbolo selezionato
#property description "Il colore e lo spessore delle barre, così come il simbolo, venç
#property description "ogni N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorBars
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- Definisce 8 colori per la colorazione delle barre (che sono memorizzate nella ma
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int      N=5;              // Il numero di ticks per cambiare il tipo
```

```
input int      bars=500;         // Il numero di barre da mostrare
input bool     messages=false;   // Mostra i messaggi nel log "Expert Advisors"
//--- Buffer Indicatore
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- Nome simbolo
string symbol;
int    bars_colors;
//--- Un array per memorizzare i colori contiene 14 elementi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
//---- Numero di colori per le barre da colorare
   bars_colors=8;   //  mostra un commento a #property indicator_color1 property
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Conta ticks per cambiare lo stile, il colore e la larghezza della barra
```

```
   ticks++;
//--- Se un numero sufficiente di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Imposta un nuovo simbolo dalla finestra Market Watch
      symbol=GetRandomSymbolName();
      //--- Cambia le proprietà della linea
      ChangeLineAppearance();
      //--- Cambia i colori usati per disegnare le candele
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- Fa 5 tentativi per riempire il buffer con i prezzi dal simbolo
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- Un contatore delle chiamate della funzione CopyFromSymbolToBuffers()
         tries++;
        }
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie i buffer indicatore con i prezzi                                           |
//+------------------------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- Nell'array rates[], copieremo Open, High, Low e Close
   MqlRates rates[];
//--- Il contatore dei tentativi
   int attempts=0;
//--- Quanto è stato copiato
   int copied=0;
//--- Fa 25 tentativi per ottenere una timeseries sul simbolo desiderato
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- Se non è riuscito a copiare un numero sufficiente di barre
   if(copied!=bars)
     {
      //--- Forma un messaggio stringa
      string comm=StringFormat("Per il simbolo %s, è riuscito a ricevere solo %d barre
                               name,
                               copied,
                               bars
```

```
                                    );
      //--- Mostra un messaggio in un commento nella finestra del grafico principale
      Comment(comm);
      //--- Mostra il messaggio
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Imposta il display del simbolo
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- Inizializza i buffers con valori vuoti
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- Copia prezzi nei buffers
   for(int i=0;i<copied;i++)
     {
      //--- Calcola l'indice appropriato per i buffers
      int buffer_index=total-copied+i;
      //--- Scrive i prezzi nei buffers
      ColorBarsBuffer1[buffer_index]=rates[i].open;
      ColorBarsBuffer2[buffer_index]=rates[i].high;
      ColorBarsBuffer3[buffer_index]=rates[i].low;
      ColorBarsBuffer4[buffer_index]=rates[i].close;
      //---
      ColorBarsColors[buffer_index]=i%bar_colors;
     }
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Restituisce in modo casuale il simoblo dal Market Watch                      |
//+------------------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Il numero dei simboli mostrati nella finestra Market Watch
   int symbols=SymbolsTotal(true);
//--- La posizione di un simbolo nella lista - un numero casuale da 0 a simboli
   int number=MathRand()%symbols;
//--- Restituisce il nome di un simbolo nella posizione specificata
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti zigzag                                         |
//+------------------------------------------------------------------------------+
```

```
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione dell'intero
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scriviar
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Cambia l'apparenza delle barre                                               |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione delle informazioni riguardo le proprietà della ba
   string comm="";

//--- Un blocco per il cambio della larghezza delle barre
   int number=MathRand();
//--- Ottiene la larghezza del resto della divisione intera
   int width=number%5;   // La larghezza è impostata da 0 a 4
//--- Imposta il colore come proprietà PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Scrive lo spessore della linea
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Scrive il nome del simbolo
   comm="\r\n"+symbol+comm;

//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

DRAW_COLOR_CANDLES

Lo stile DRAW_COLOR_CANDLES, come DRAW_CANDLES, disegna candele utilizzando i valori dei quattro buffer di indicatori, che contengono i prezzi Open, High, Low e Close. Inoltre, esso permette di specificare un colore per ogni candela da un dato insieme. A questo scopo, lo stile ha un colore speciale buffer che memorizza indici di colore per ogni barra. E' utilizzato per la creazione di indicatori personalizzati come una sequenza di candele, comprese quelle in una sottofinestra separata di un grafico e su altri strumenti finanziari.

Il numero di colori delle candele può essere impostato utilizzando le direttive del compilatore o in modo dinamico utilizzando la funzione PlotIndexSetInteger(). Cambiamenti dinamici delle proprietà di plotting permettono di "animare" gli indicatori, in modo che il loro aspetto cambia a seconda della situazione attuale.

L'indicatore è disegnato solo per quelle barre per cui sono impostati valori non vuoti di quattro buffer di prezzi dell'indicatore. Per specificare quale valore deve essere considerato come "vuoto", impostare questo valore nella proprietà PLOT_EMPTY_VALUE:

```
/ / --- Il valore 0 (vuoto) parteciperà al disegno
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,0);
```

Riempie esplicitamente i valori dei buffer indicatori, imposta un valore vuoto in un buffer per saltare le barre.

Il numero di buffer necessari per il plotting di DRAW_COLOR_CANDLES è di 5:

- quattro buffer per memorizzare i valori di Open, High, Low e Close;
- un buffer per memorizzare l'indice del colore, che viene utilizzato per disegnare una candela (ha senso impostarlo solo per le candele che verranno disegnate).

Tutti i buffer per il tracciamento dovrebbero andare uno dopo l'altro, nell'ordine: Open, High, Low, Close e il buffer del colore. Nessuno dei buffer di prezzo possono contenere solo valori vuoti, poiché in questo caso nulla è disegnato.

Un esempio di indicatore che disegna candele per uno strumento finanziario selezionato in una finestra separata. Il colore delle candele cambia dinamicamente ogni **N** ticks. Il parametro N è impostato nei parametri esterni dell'indicatore per la possibilità di configurazione manuale (la scheda Parametri nella finestra Proprietà dell' indicatore).

Si prega di notare che per **plot1**, il colore è impostato usando la direttiva del compilatore #property, e poi nella funzione OnCalculate() il colore è impostato in modo casuale da una lista precedentemente preparata.

```
//+------------------------------------------------------------------------------------+
//|                                                          DRAW_COLOR_CANDLES.mq5 |
//|                                       Copyright 2011, MetaQuotes Software Corp. |
//|                                                             https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicatore per dimostrare DRAW_COLOR_CANDLES."
#property description "Disegna le candele di un simbolo selezionato in una finestra se
#property description " "
#property description "Il colore e lo spessore delle candele, così come il simbolo ver
#property description "dinamicamente ogni N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- Definisce 8 colori per colorare le candele (sono memorizzati nell'array speciale
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//--- parametri di input
input int      N=5;              // Il numero di ticks per cambiare il tipo
input int      bars=500;         // Il numero di candele da mostrare
input bool     messages=false;   // Mostra i messaggi nel log "Expert Advisors"
//--- Buffer Indicatore
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- Nome simbolo
string symbol;
//--- Un array per memorizzare i colori contiene 14 elementi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
/ / --- Se bars è molto piccolo - completa il lavoro prima del tempo
   if(bars<50)
     {
      Comment("Prego specificare un gran numero di barre! L'operazione dell'indicatore
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- Un valore vuoto
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Il nome del simbolo, per cui le barre vengono disegnate
   symbol=_Symbol;
//--- Imposta il display del simbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- Il numero di colori di candele colore
   candles_colors=8;     //  see. un commento alla proprietà #property indicator_colo
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione di iterazione indicatore personalizato                          |
//+--------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- Conteggio di ticks per cambiare lo stile ed il colore
   ticks++;
//--- Se un numero sufficiente di ticks è stato accumulato
   if(ticks>=N)
     {
      //--- Imposta un nuovo simbolo dalla finestra Market Watch
      symbol=GetRandomSymbolName();
      //--- Cambia la forma
      ChangeLineAppearance();
      //--- Cambia i colori usati per disegnare le candele
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- Fa 5 tentativi per riempire i buffers di plot1 con i prezzi dal simbolo
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- Un contatore delle chiamate della funzione CopyFromSymbolToBuffers()
         tries++;
        }
      //--- Resetta il contatore dei ticks a zero
      ticks=0;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+--------------------------------------------------------------------------+
//| Riempie le candele specificate                                           |
//+--------------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

```
                              double &buff2[],
                              double &buff3[],
                              double &buff4[],
                              double &col_buffer[],
                              int    cndl_colors
                              )
  {
//--- Nell'array rates[], copieremo Open, High, Low e Close
   MqlRates rates[];
//--- Il contatore dei tentativi
   int attempts=0;
//--- Quanto è stato copiato
   int copied=0;
//--- Fa 25 tentativi per ottenere una timeseries sul simbolo desiderato
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- Se non è riuscito a copiare un numero sufficiente di barre
   if(copied!=bars)
     {
      //--- Forma un messaggio stringa
      string comm=StringFormat("Per il simbolo %s, è riuscito a ricevere solo %d barre
                               name,
                               copied,
                               bars
                               );
      //--- Mostra un messaggio in un commento nella finestra del grafico principale
      Comment(comm);
      //--- Mostra il messaggio
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Imposta il display del simbolo
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- Inizializza i buffers con valori vuoti
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- Su ogni tick, copia i prezzi dei buffers
   for(int i=0;i<copied;i++)
     {
```

```
      //--- Calcola l'indice appropriato per i buffers
      int buffer_index=total-copied+i;
      //--- Scrive i prezzi nei buffers
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- Imposta il colore della candela
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Restituisce in modo casuale il simoblo dal Market Watch                      |
//+------------------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Il numero dei simboli mostrati nella finestra Market Watch
   int symbols=SymbolsTotal(true);
//--- La posizione di un simbolo nella lista - un numero casuale da 0 a simboli
   int number=MathRand()%symbols;
//--- Restituisce il nome di un simbolo nella posizione specificata
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------------------+
//| Cambia il colore dei segmenti delle candele                                  |
//+------------------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Il numero dei colori
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Per ogni indice colore definisce un nuovo colore casualmente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Ottiene un valore casuale
      int number=MathRand();
      //--- Ottiene un indice nell'array col[] come resto della divisione dell'intero
      int i=number%size;
      //--- Imposta il colore per ogni indice che ha proprietà PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  Il numero di stili grafici
                          PLOT_LINE_COLOR,      //  Identificatore proprietà
                          plot_color_ind,       //  Liindice del colore, dove scriviar
                          cols[i]);             //  Un nuovo colore
      //--- Scrivi i colori
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
```

```
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------------------+
//| Cambia l'apparenza delle candele                                             |
//+------------------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Una stringa per la formazione di informazioni sulle proprietà delle candele
   string comm="";
//--- Scrive il nome del simbolo
   comm="\r\n"+symbol+comm;
//--- Imposta le informazioni del grafico usando un commento
   Comment(comm);
  }
```

Correlazione tra Proprietà Indicatore e Funzioni Corrispondenti

Ogni indicatore personalizzato ha numerose proprietà, alcune delle quali sono obbligatorie e sono sempre posizionate all'inizio della descrizione. Esse sono le seguenti:

- indicazione di una finestra per tracciare l'indicatore - indicator_separate_window o indicator_chart_window;
- numero di buffer indicatore - indicator_buffers;
- numero di plots dell' indicatore - indicator_plots.

Inoltre ci sono altre proprietà che possono essere impostate sia attraverso le direttive del preprocessore che attraverso funzioni intese per la creazione di indicatore personalizzato. Queste proprietà e funzioni corrispondenti vengono descritte nella seguente tabella.

Direttive per le proprietà della sottofinestra indicatore	Funzioni di tipo IndicatorSet ... ()	Descrizione della proprietà rettificata della sottofinestra
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	Il valore fisso dell'altezza della sottofinestra
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	Valore minimo dell'asse verticale
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	Valore massimo dell'asse verticale
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	Valore dell'asse verticale per il livello di N
nessuna direttiva del preprocessore	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	Nome di un livello visualizzato
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	Colore del livello N
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	Larghezza della linea per il livello N
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	Stile di linea per il livello N
Direttive per le proprietà plottaggio	**Funzioni di tipo PlotIndexSet ... ()**	**Descrizione della proprietà rettificata di un plot**

Direttive per le proprietà della sottofinestra indicatore	Funzioni di tipo IndicatorSet ... ()	Descrizione della proprietà rettificata della sottofinestra
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	Nome breve del numero N di plot. Viene visualizzato in DataWindow e nella descrizione comandi a comparsa quando si punta il cursore del mouse su di essa
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	Colore linea per il plot N
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	Stile di linea per il plot N
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	Tipo di linea per il plot N
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	Larghezza della linea per il plot N
Proprietà comuni degli indicatori	**Funzioni di tipo IndicatorSet ... ()**	**Descrizione**
nessuna direttiva del preprocessore	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	Imposta il conveniente nome breve dell'indicatore che sarà visualizzato nella lista degli indicatori (aperto nel terminale premendo **Ctrl+I**).
nessuna direttiva del preprocessore	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	Imposta l'accuratezza di visualizzazione richiesta dei valori degli indicatori - numero di cifre decimali
nessuna direttiva del preprocessore	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	Imposta il numero di livelli nella finestra dell' indicatore
indicator_applied_price	Nessuna funzione, la proprietà può essere impostata solo con la direttiva del preprocessore.	Il tipo di prezzo di default utilizzato per il calcolo dell'indicatore. E' specificato quando necessario, solo utilizzando OnCalculate() del primo tipo. Il valore della proprietà può essere impostato anche dalla finestra di dialogo delle proprietà dell' indicatore nella scheda "Parametri" - "Applicare a".

Va notato che la numerazione dei livelli e plots in termini preprocessore inizia con uno, mentre la numerazione delle stesse proprietà utilizzando le funzioni inizia con zero, cioè il valore indicato deve essere di 1 minore di quello indicato per #property.

Ci sono diverse direttive, per le quali non vi sono funzioni corrispondenti:

Direttiva	Descrizione
indicator_chart_window	L' indicatore viene visualizzato nella finestra principale
indicator_separate_window	L' indicatore viene visualizzato in una finestra secondaria separata
indicator_buffers	Numero di indicatori buffer richiesti
indicator_plots	Numero di plots nell'indicatore

SetIndexBuffer

La funzione associa un buffer indicatore specificato con l'array dinamico unidimensionale di tipo double.

```
bool  SetIndexBuffer(
   int                   index,         // indice buffer
   double                buffer[],      // array
   ENUM_INDEXBUFFER_TYPE data_type      // ciò che verrà memorizzato
   );
```

Parametri

index

[in] Numero del buffer dell'indicatore. La numerazione inizia da 0. Il numero deve essere inferiore al valore dichiarato in #property indicator_buffers.

buffer[]

[in] Un array dichiarato nel programma indicatore personalizzato.

data_type

[in] tipo di dati memorizzati nell' array indicatore. Per impostazione predefinita è INDICATOR_DATA (valori dell'indicatore calcolato). Si può anche prendere il valore di INDICATOR_COLOR_INDEX; in questo caso tale buffer è usato per memorizzare indici di colore per il buffer indicatore precedente. È possibile specificare fino a 64 colori nella linea #property indicator_colorN. I valori INDICATOR_CALCULATIONS indicano che il buffer è utilizzato nei calcoli intermedi di indicatore dell'indicatore e non è inteso per il disegno.

Valore restituito

In caso di successo, restituisce true, altrimenti - false.

Nota

Dopo il legame, il buffer[] dell'array dinamico verrà indicizzato come nei comuni array, anche se l'indicizzazione delle timeseries è pre-installata per l'array legato. Se si desidera modificare l'ordine di accesso agli elementi dell'array indicatore, utilizzare la funzione ArraySetAsSeries() **dopo aver legato l'array utilizzando la funzione SetIndexBuffer()**. Tenere presente che non è possibile modificare la dimensione di array dinamici impostati come buffer indicatore dalla funzione SetIndexBuffer(). Per i buffer indicatore, tutte le operazioni di cambio grandezza vengono svolti dal sottosistema di esecuzione del terminale.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                      TestCopyBuffer1.mq5 |
//|                                  Copyright 2009, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input bool               AsSeries=true;
input int                period=15;
input ENUM_MA_METHOD     smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int                shift=0;
//--- buffers indicatore
double                   MABuffer[];
int                      ma_handle;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("Il buffer indicatore è una timeseries = ",ArrayGetAsSeries(MABuffer));
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Il buffer indicatore dopo SetIndexBuffer() è una timeseries = ",
         ArrayGetAsSeries(MABuffer));

//--- cambia l'ordine di accesso agli elementi del buffer indicatore
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
                const long &volume[],
                const int &spread[])
  {
//--- Copia i valori della media mobile nel buffer MABuffer
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// Dipende dai valori di AsSeries
                                       // Riceverà un valore molto vecchio
                                       // O per la corrente barra non finita

//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
```

Vedi anche

Proprietà Indicatori Personalizzati, Accesso alle timeseries ed indicatori

IndicatorSetDouble

La funzione imposta il valore della proprietà indicatore corrispondente. Le proprietà dell'indicatore deve essere di tipo double. Ci sono due varianti della funzione.

Chiamare specificando l'identificatore di proprietà.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificatore
   double  prop_value         // valore da impostare
   );
```

Chiamare specificando l'identificatore della proprietà ed il modificatore.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificatore
   int     prop_modifier,     // modificatore
   double  prop_value         // valore da impostare
   )
```

Parametri

prop_id

[in] Identificatore della proprietà dell'indicatore. Il valore può essere uno dei valori dell'enumerazione ENUM_CUSTOMIND_PROPERTY_DOUBLE.

prop_modifier

[in] Modificatore della proprietà specificata. Solo le proprietà livello richiedono un modificatore. La numerazione dei livelli parte da 0. Ciò significa che, al fine di impostare proprietà per il secondo livello è necessario specificare 1 (1 in meno rispetto a quando si utilizza la direttiva del compilatore).

prop_value

[in] Valore della proprietà.

Valore restituito

In caso di esecuzione avvenuta con successo, restituisce true, altrimenti - false

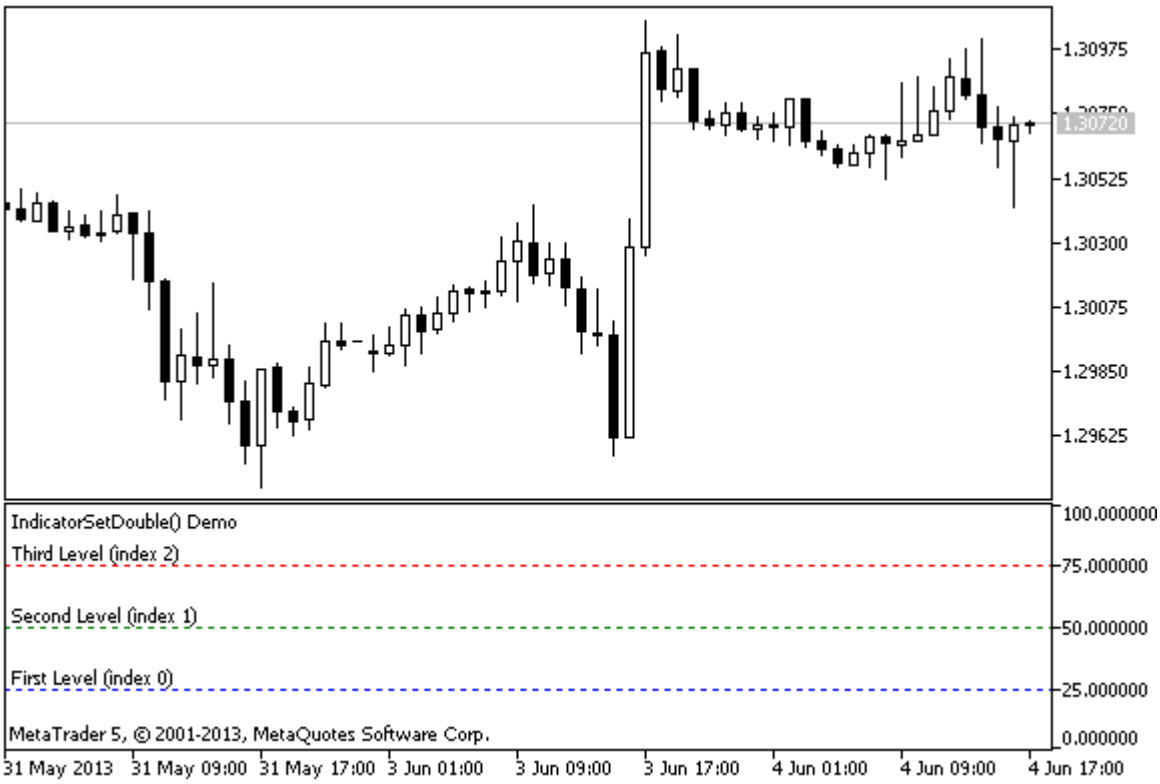
Nota

La numerazione delle proprietà (modificatori) comincia da 1 (uno) quando si utilizza la direttiva #property, mentre la funzione utilizza la numerazione da 0 (zero). Nel caso in cui il numero di livello è impostato in modo errato, la visualizzazione dell' indicatore può differire da quanto previsto.

Ad esempio, il primo valore del livello per l'indicatore in una sottofinestra separata può essere impostato in due modi:

- property indicator_level**1** 50 - il valore 1 è utilizzato per specificare il numero del livello,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) - 0 è usato per specificare il primo livello.

Esempio: indicatore che capovolge i valori massimi e minimi della finestra indicatore e valori dei livelli su cui sono posizionate le linee orizzontali.




```
#property indicator_separate_window
//--- Imposta i valori massimi e minimi per la finestra dell'indicatore
#property indicator_minimum  0
#property indicator_maximum  100
//--- visualizza tre livelli orizzontali in una finestra indicatore separata
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- imposta lo spessore dei livelli orizzontali
#property indicator_levelwidth 1
//--- imposta lo stile dei livelli orizzontali
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- imposta le descrizioni dei livelli orizzontali
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"Primo livello (indice 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Secondo livello (indice 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Terzo livello (indice 2)");
//--- imposta il nome corto per l'indicatore
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- imposta il colore per ogni livello
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
   static double level1=25,level2=50,level3=75;
   static double max=100,min=0, shift=100;
//--- calcola i ticks
   tick_counter++;
//--- capovolge i livelli sottosopra ogni 10imo tick
   if(tick_counter%10==0)
     {
      //--- inverte il segno per i valori dei livelli
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- inverte il segno per i valori massimo e minimo
      max-=shift;
      min-=shift;
      //--- inverte il valore di shift
      shift=-shift;
      //--- imposta i valori dei nuovi livelli
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- imposta i nuovi valori di massimo e minimo nella finestra indicatore
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Stili Indicatore negli Esempi, Correlazione tra Proprietà Indicatore e Funzioni, Stili di Disegno

IndicatorSetInteger

La funzione imposta il valore della proprietà indicatore corrispondente. La proprietà indicatore deve essere di int o di tipo color. Ci sono due varianti della funzione.

Chiamare specificando l'identificatore di proprietà.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificatore
   int  prop_value         // valore da impostare
   );
```

Chiamare specificando l'identificatore della proprietà ed il modificatore.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificatore
   int  prop_modifier,     // modificatore
   int  prop_value         // valore da impostare
   )
```

Parametri

prop_id

[in] Identificatore della proprietà dell'indicatore. Il valore può essere uno dei valori dell'enumerazione ENUM_CUSTOMIND_PROPERTY_INTEGER.

prop_modifier

[in] Modificatore della proprietà specificata. Solo le proprietà livello richiedono un modificatore.

prop_value

[in] Valore della proprietà.

Valore restituito

In caso di esecuzione avvenuta con successo, restituisce true, altrimenti - false

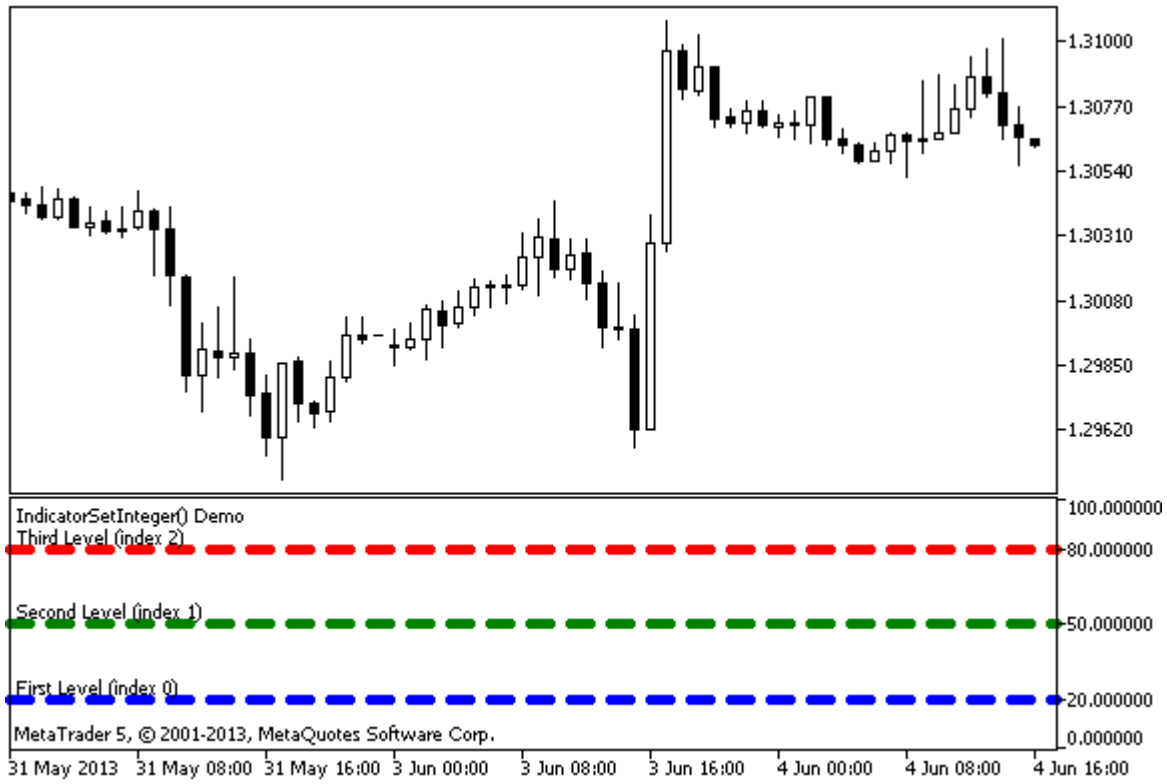
Nota

La numerazione delle proprietà (modificatori) comincia da 1 (uno) quando si utilizza la direttiva #property, mentre la funzione utilizza la numerazione da 0 (zero). Nel caso in cui il numero di livello è impostato in modo errato, la visualizzazione dell' indicatore può differire da quanto previsto.

Ad esempio, al fine di impostare lo spessore della prima linea orizzontale, usa l'indice zero:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) - l'indice 0 viene usato per impostare lo spessore del primo livello.

Esempio: Indicatore che imposta il colore, lo stile e lo spessore delle linee orizzontali dell'indicatore.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- visualizza tre livelli orizzontali in una finestra indicatore separata
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- imposta lo spessore dei livelli orizzontali
#property indicator_levelwidth 5
//--- imposta il colore dei livelli orizzontali
#property indicator_levelcolor clrAliceBlue
//--- imposta lo stile dei livelli orizzontali
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- imposta le descrizioni dei livelli orizzontali
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"Primo livello (indice 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Secondo livello (indice 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Terzo livello (indice 2)");
//--- imposta il nome corto per l'indicatore
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione di iterazione indicatore personalizato                          |
//+--------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
//--- calcola i ticks
   tick_counter++;
//--- e calcola i colori dei livelli orizzontali a seconda del contatore del tick
   ChangeLevelColor(0,tick_counter,3,6,10); // gli ultimi tre paramteri cambiano colo
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- modifica lo stile delle linee orizzontali
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- ottiene lo spessore come resto della divisione intera del numero di ticks per 5
   int width=tick_counter%5;
//--- iterazione su tutti i livelli orizzontali ed impostazioni dello spessore
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+--------------------------------------------------------------------------+
//| Imposta il colore della linea orizzontale nella finestra indicatore separata |
//+--------------------------------------------------------------------------+
void ChangeLevelColor(int level,      // numero di linee orizzontali
                      int tick_number,// dividendo, numero per ottenere il resto della
                      int f_trigger,  // primo divisore del cambio di colore
                      int s_trigger,  // secondo divisore del cambio di colore
                      int t_trigger)  // terzo divisore del cambio di colore
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- indice del colore dall'array colors[]
   int index=-1;
//--- calcola il numero di colori dall'array colors[] per disegnare una linea orizzont
   if(tick_number%f_trigger==0)
      index=0;   // se tick_number è diviso per f_trigger senza dare resto
   if(tick_number%s_trigger==0)
      index=1;   // se tick_number è diviso per s_trigger senza dare resto
```

```
   if(tick_number%t_trigger==0)
      index=2;    // se tick_number è diviso per t_trigger senza dare resto
//--- se il colore è definito, impostarlo
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------------------+
//| Imposta lo stile della linea orizzontale nella finestra indicatore separata  |
//+------------------------------------------------------------------------------+
void ChangeLevelStyle(int level,     // numero della linea orizzontale
                      int tick_number// numer per ottenere il resto della divisione
                      )
  {
//--- array per memorizzare gli stili
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- indice degli stili dall'array styles[]
   int index=-1;
//--- calcola il numero dall'array styles[] per impostare lo stile della linea orizzon
   if(tick_number%50==0)
      index=5;    // se tick_number diviso per 50 è senza resto, allora lo stile è STYL
   if(tick_number%40==0)
      index=4;    // ... lo stile è STYLE_DASHDOT
   if(tick_number%30==0)
      index=3;    // ... STYLE_DOT
   if(tick_number%20==0)
      index=2;    // ... STYLE_DASH
   if(tick_number%10==0)
      index=1;    // ... STYLE_SOLID
//--- se lo stile è definito, impostarlo
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

Vedi anche

Proprietà Indicatori Personalizzati, Proprietà Programmi (#property), Stili di disegno

IndicatorSetString

La funzione imposta il valore della proprietà indicatore corrispondente. La proprietà indicatore deve essere di tipo string. Ci sono due varianti della funzione.

Chiamare specificando l'identificatore di proprietà.

```
bool  IndicatorSetString(
   int     prop_id,           // identificatore
   string  prop_value         // valore da impostare
   );
```

Chiamare specificando l'identificatore della proprietà ed il modificatore.

```
bool  IndicatorSetString(
   int     prop_id,           // identificatore
   int     prop_modifier,     // modificatore
   string  prop_value         // valore da impostare
   )
```

Parametri

prop_id

[in] Identificatore della proprietà dell'indicatore. Il valore può essere uno dei valori dell'enumerazione ENUM_CUSTOMIND_PROPERTY_STRING.

prop_modifier

[in] Modificatore della proprietà specificata. Solo le proprietà livello richiedono un modificatore.

prop_value

[in] Valore della proprietà.

Valore restituito

In caso di esecuzione avvenuta con successo, restituisce true, altrimenti - false

Nota

La numerazione delle proprietà (modificatori) comincia da 1 (uno) quando si utilizza la direttiva #property, mentre la funzione utilizza la numerazione da 0 (zero). Nel caso in cui il numero di livello è impostato in modo errato, la visualizzazione dell' indicatore può differire da quanto previsto.

Ad esempio, per impostare la descrizione della prima linea orizzontale, usa l'indice zero:

- IndicatorSetString(INDICATOR_LEVELTEXT, **0**, "First Level") - l'indice 0 viene usato per impostare la descrizione del testo del primo livello.

Esempio: indicatore che imposta l'etichetta del testo sulla linea orizzontale dell'indicatore.


```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- visualizza tre livelli orizzontali in una finestra indicatore separata
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- imposta il colore dei livelli orizzontali
#property indicator_levelcolor clrRed
//--- imposta lo stile dei livelli orizzontali
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- imposta le descrizioni dei livelli orizzontali
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"Primo livello (indice 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Secondo livello (indice 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Terzo livello (indice 2)");
//--- imposta il nome corto per l'indicatore
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizzato                 |
```

```
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---

//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
```

Vedi anche

Proprietà Indicatori Personalizzati, Proprietà Programmi (#property)

PlotIndexSetDouble

La funzione imposta il valore della proprietà corrispondente della corrispondente linea dell'indicatore. La proprietà indicatore deve essere di tipo double.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // indice di stile di plotting
   int     prop_id,        // identificatore proprietà
   double  prop_value      // valore da impostare
   );
```

Parametri

plot_index

[in] Index of the graphical plotting

prop_id

[in] Il valore può essere uno dei valori dell'enumerazione ENUM_PLOT_PROPERTY_DOUBLE.

prop_value

[in] Il valore della proprietà.

Valore restituito

In caso di successo, restituisce true, altrimenti false.

PlotIndexSetInteger

La funzione imposta il valore della proprietà corrispondente della corrispondente linea dell'indicatore. La proprietà indicatore deve essere di tipo int, char, bool o color,. Ci sono due varianti della funzione.

Chiamare indicando identificatore della proprietà.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // indice dello stile di plotting
   int  prop_id,            // identificatore proprietà
   int  prop_value          // valore da impostare
   );
```

Chiamare indicando l'identificatore e modificatore della proprietà.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // indice dello stile di plotting
   int  prop_id,            // identificatore proprietà
   int  prop_modifier,      // modificatore proprietà
   int  prop_value          // valore da impostare
   )
```

Parametri

plot_index

[in] Index of the graphical plotting

prop_id

[in] Il valore può essere uno dei valori dell'enumerazione ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Modificatore della proprietà specificata. Solo proprietà di indice di colore richiedono un modificatore.

prop_value

[in] Il valore della proprietà.

Valore restituito

In caso di successo, restituisce true, altrimenti false.

Esempio: un indicatore che disegna una linea a tre colori. Lo schema dei colori cambia ogni 5 ticks.


```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrGreen,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- buffers indicatore
double         ColorLineBuffer[];
double         ColorBuffer[];
int            MA_handle;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato           |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- get MA handle
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| ottieni l'indice del colore                                      |
//+------------------------------------------------------------------+
```

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// primo indice
   if(j<200) return(1);// secondo indice
   return(2); // terzo indice
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   static int ticks=0,modified=0;
   int limit;
//--- il primo calcolo o il numero delle barre è stato cambiato
   if(prev_calculated==0)
     {
      //--- copia valori di MA nel buffer indicatore ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// copying failed - throw away
      //--- ora imposta il colore della linea per ogni barra
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- copia valori di MA nel buffer indicatore ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// conteggio ticks
      if(ticks>=5)//è ora di cambiare lo schema del colore
        {
         ticks=0; // resetta il contatore
         modified++; // contatore dei cambiamenti del colore
         if(modified>=3)modified=0;// resetta il contatore
         ResetLastError();
         switch(modified)
           {
```

```
            case 0:// schema primo colore
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Schema Colore"+modified);
               break;
            case 1:// schema secondo colore
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Schema Colore"+modified);
               break;
            default:// schema terzo colore
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Schema Colore"+modified);
           }
        }
      else
        {
         //--- imposta la posizione di inizio
         limit=prev_calculated-1;
         //--- ora impostiamo il colore della linea per ogni barra
         for(int i=limit;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
        }
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
```

PlotIndexSetString

La funzione imposta il valore della proprietà corrispondente della corrispondente linea dell'indicatore. La proprietà indicatore deve essere di tipo string.

```
bool  PlotIndexSetString(
   int     plot_index,     // indice di stile di plotting
   int     prop_id,        // identificatore proprietà
   string  prop_value      // valore da impostare
   );
```

Parametri

plot_index

[in] Indice del graphical plot

prop_id

[in] Il valore può essere uno dei valori dell' enumerazione ENUM_PLOT_PROPERTY_STRING.

prop_value

[in] Il valore della proprietà.

Valore restituito

In caso di successo, restituisce true, altrimenti false.

PlotIndexGetInteger

La funzione imposta il valore della proprietà corrispondente della corrispondente linea dell'indicatore. La proprietà indicatore deve essere del tipo int, color, bool o char. Ci sono due varianti della funzione.

Chiamare indicando identificatore della proprietà.

```
int  PlotIndexGetInteger(
   int  plot_index,         // indice dello stile di plotting
   int  prop_id,            // identificatore proprietà
   );
```

Chiamare indicando l'identificatore e modificatore della proprietà.

```
int  PlotIndexGetInteger(
   int  plot_index,         // indice plotting
   int  prop_id,            // identificatore proprietà
   int  prop_modifier       // modificatore proprietà
   )
```

Parametri

plot_index

[in] Index of the graphical plotting

prop_id

[in] Il valore può essere uno dei valori dell'enumerazione ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Modificatore della proprietà specificata. Solo proprietà di indice di colore richiedono un modificatore.

Nota

La funzione è progettata per estrarre le impostazioni del disegno della linea indicatore appropriata. La funzione lavora in tandem con la funzione PlotIndexSetInteger per copiare le proprietà del disegno di una linea all'altra.

Esempio: un indicatore che colora le candele a seconda del giorno della settimana. I colori per ogni giorno vengono impostati in modo programma.


```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers indicatore
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
void OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- imposta il numero di colori nel buffer color
```

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- imposta i colori nel buffer color
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- imposta precisione
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   int i;
   MqlDateTime t;
//----
   if(prev_calculated==0) i=0;
   else i=prev_calculated-1;
//----
   while(i<rates_total)
     {
      OpenBuffer[i]=open[i];
      HighBuffer[i]=high[i];
      LowBuffer[i]=low[i];
      CloseBuffer[i]=close[i];
      //--- imposta il colore per ogni candela
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
```

Oggetti Grafici

Questo è il gruppo di funzioni destinate a lavorare con oggetti grafici relativi a qualsiasi grafico specificato.

Le funzioni che definiscono le proprietà degli oggetti grafici, nonché le operazioni ObjectCreate() e ObjectMove() per creare e spostare gli oggetti lungo il chart vengono effettivamente utilizzate per l'invio di comandi al chart. Se queste funzioni vengono eseguite correttamente, il comando viene incluso nella coda comune degli eventi chart. Alterazioni visive nelle proprietà degli oggetti grafici vengono implementate quando si maneggia la coda degli eventi del grafico.

Quindi, non aspettatevi un aggiornamento visivo immediato di oggetti grafici dopo la chiamata di queste funzioni. In generale, gli oggetti grafici sul chart vengono aggiornati automaticamente dal terminale in seguito agli eventi di cambiamento - un nuovo arrivo di quotazione, il ridimensionamento della finestra chart, ecc. Usare la funzione ChartRedraw() per aggiornare con forza gli oggetti grafici.

Funzione	Azione
ObjectCreate	Crea un oggetto del tipo specificato in una tabella specificata
ObjectName	Restituisce il nome di un oggetto del tipo corrispondente nella tabella specificata (grafico sottofinestra specificato)
ObjectDelete	Rimuove l'oggetto con il nome specificato dalla tabella specificata (dalla sottofinestra grafico specificata)
ObjectsDeleteAll	Rimuove tutti gli oggetti del tipo specificato dalla tabella specificata (dalla sottofinestra grafico specificata)
ObjectFind	Cerca un oggetto con l'ID specificato, per nome
ObjectGetTimeByValue	Restituisce il valore di tempo per il valore dell'oggetto prezzo specificato
ObjectGetValueByTime	Restituisce il valore del prezzo di un oggetto per il periodo di tempo specificato
ObjectMove	Cambia le coordinate del punto di ancoraggio dell' oggetto specificato
ObjectsTotal	Restituisce il numero di oggetti del tipo specificato nella tabella specificata (grafico sottofinestra specificato)
ObjectGetDouble	Restituisce il doppio valore della proprietà dell'oggetto corrispondente
ObjectGetInteger	Restituisce il valore intero della proprietà dell'oggetto corrispondente
ObjectGetString	Restituisce il valore stringa della proprietà dell'oggetto corrispondente
ObjectSetDouble	Imposta il valore della proprietà dell'oggetto corrispondente
ObjectSetInteger	Imposta il valore della proprietà dell'oggetto corrispondente

Funzione	Azione
ObjectSetString	Imposta il valore della proprietà dell'oggetto corrispondente
TextSetFont	Consente di impostare il tipo di carattere per la visualizzazione del testo utilizzando metodi di disegno (Arial 20 utilizzato per impostazione predefinita)
TextOut	Trasferisce il testo ad un array personalizzato (buffer) progettato per la creazione di una risorsa grafica
TextGetSize	Restituisce la larghezza della stringa e l'altezza alle correnti Impostazioni dei caratteri

Ogni oggetto grafico deve avere un nome univoco all'interno di un grafico, comprese le suoe sottofinestre. La modifica di un nome di un oggetto grafico genera due eventi: evento di eliminazione di un oggetto con il vecchio nome, ed eventi di creazione di un oggetto con un nuovo nome.

Dopo che un oggetto viene creato o una proprietà di un oggetto viene modificata, si consiglia di chiamare la funzione ChartRedraw(), che comanda il terminale client per disegnare forzatamente un grafico (e tutti gli oggetti visibili in esso contenuti).

ObjectCreate

La funzione crea un oggetto con il nome, il tipo e le coordinate iniziale specificati, nella sottofinestra grafico specificata. Durante la creazione, fino a 30 coordinate possono essere specificate.

```
bool  ObjectCreate(
   long          chart_id,      // identificatore del grafico
   string        name,          // nome dell'oggetto
   ENUM_OBJECT   type,          // tipo dell'oggetto
   sub_window    nwin,          // indice della finestra
   datetime      time1,         // tempo per il primo punto di ancoraggio
   double        price1,        // prezzo per il primo punto di ancoraggio
   ...
   datetime      timeN=0,       // tempo dell' N-simo punto di ancoraggio
   double        priceN=0,      // prezzo tempo dell' N-simo punto di ancoraggio
   ...
   datetime      time30=0,      // tempo del 30esimo punto di ancoraggio
   double        price30=0      // prezzo del 30esimo punto di ancoraggio
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto. Il nome deve essere univoco all'interno di un grafico, incluse le sue sottofinestre.

type

[in] Tipo dell' Oggetto. Il valore può essere uno dei valori dell'enumerazione ENUM_OBJECT.

sub_window

[in] Numero di sottofinestra grafico. 0 significa la finestra principale del grafico. La sottofinestra specificata deve esistere, altrimenti la funzione restituisce false.

time1

[in] La coordinata temporale del primo ancoraggio.

price1

[in] La coordinata prezzo del primo punto di ancoraggio.

timeN=0

[in] La coordinata temporale dell' N-esimo punto di ancoraggio.

priceN=0

[in] La coordinata prezzo dell' N-esimo punto di ancoraggio.

time30=0

[in] coordinata temporale del trentesimo punto di ancoraggio.

price30=0

[in] La coordinata prezzo del trentesimo punto di ancoraggio.

Valore restituito

La funzione restituisce true se il comando è stato aggiunto con successo alla coda del chart specificato, altrimenti false. Se un oggetto è già stato creato, si tenta di modificare le coordinate.

Nota

Una chiamata asincrona viene sempre utilizzata per ObjectCreate(), che è ciò per cui la funzione restituisce solo i risultati di aggiunta comando ad una coda(queue) chart. In questo caso, true significa solo che il comando è stato accodato correttamente, ma il risultato della sua esecuzione non è noto.

Per verificare il risultato di esecuzione del comando, è possibile utilizzare la funzione ObjectFind() o qualsiasi altra funzione che richiede proprietà dell'oggetto, come ObjectGetXXX. Tuttavia, dovresti tenere presente che tali funzioni vengono aggiunte alla fine della coda di quel grafico ed aspettano il risultato dell'esecuzione (dovuto alla chiamata sincrona) e quindi possono richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Un nome di oggetto non deve superare i 63 caratteri.

La numerazione delle sottofinestre grafico (se ci sono sotto-finestre con indicatori nella tabella) inizia con 1. La finestra grafico principale del grafico è, e sempre ha, indice 0.

Il gran numero di punti di ancoraggio (fino a 30) è implementato per uso futuro. At stesso tempo, il limite di 30 punti di ancoraggio possibili per gli oggetti grafici è determinato dal limite al numero di parametri (non più di 64) che può essere utilizzato quando si chiama una funzione.

Quando un oggetto viene rinominato, due eventi si formano simultaneamente. Questi eventi possono essere gestiti in un Expert Advisor o indicatore della funzione OnChartEvent():

- un evento di eliminazione di un oggetto con il vecchio nome;
- un evento di creazione di un oggetto con un nuovo nome.

Vi è un certo numero di punti di ancoraggio che devono essere specificati al momento della creazione di ogni Tipo di oggetto:

ID	Descrizione	Punti di ancoraggio
OBJ_VLINE	Linea verticale	Un punto di ancoraggio. Al momento, solo la coordinata temporale viene utilizzata.
OBJ_HLINE	Linea orizzontale	Un punto di ancoraggio. Al momento solo la coordinata prezzo viene utilizzata.
OBJ_TREND	Trend Line	Due punti di ancoraggio.
OBJ_TRENDBYANGLE	Trend Line per Angolo	Due punti di ancoraggio.
OBJ_CYCLES	Linee Cicliche	Due punti di ancoraggio.
OBJ_ARROWED_LINE	Linea con freccia	Due punti di ancoraggio.

ID	Descrizione	Punti di ancoraggio
OBJ_CHANNEL	Canale Equidistante	Tre punti di ancoraggio.
OBJ_STDDEVCHANNEL	Canale Deviazione Standard	Due punti di ancoraggio.
OBJ_REGRESSION	Canale Regressione Lineare	Due punti di ancoraggio.
OBJ_PITCHFORK	Andrews' Pitchfork	Tre punti di ancoraggio.
OBJ_GANNLINE	La Linea Gann	Due punti di ancoraggio.
OBJ_GANNFAN	In ventaglio Gann	Due punti di ancoraggio.
OBJ_GANNGRID	La griglia Gann	Due punti di ancoraggio.
OBJ_FIBO	Ritracciamento di Fibonacci	Due punti di ancoraggio.
OBJ_FIBOTIMES	Le Fibonacci Time Zones	Due punti di ancoraggio.
OBJ_FIBOFAN	Il ventaglio Fibonacci	Due punti di ancoraggio.
OBJ_FIBOARC	L' arco Fibonacci	Due punti di ancoraggio.
OBJ_FIBOCHANNEL	Il canale Fibonacci	Tre punti di ancoraggio.
OBJ_EXPANSION	L'espansione Fibonacci	Tre punti di ancoraggio.
OBJ_ELLIOTWAVE5	L'onda motiva Elliott	Cinque punti di ancoraggio.
OBJ_ELLIOTWAVE3	L'onda correttiva Elliott	Tre punti di ancoraggio.
OBJ_RECTANGLE	Rettangolo	Due punti di ancoraggio.
OBJ_TRIANGLE	Triangolo	Tre punti di ancoraggio.
OBJ_ELLIPSE	Ellisse	Tre punti di ancoraggio.
OBJ_ARROW_THUMB_UP	Pollice su	Un punto di ancoraggio.
OBJ_ARROW_THUMB_DOWN	Pollice giu	Un punto di ancoraggio.
OBJ_ARROW_UP	Freccia su	Un punto di ancoraggio.
OBJ_ARROW_DOWN	Freccia giu	Un punto di ancoraggio.
OBJ_ARROW_STOP	Segno di Stop	Un punto di ancoraggio.
OBJ_ARROW_CHECK	Segno di Visto	Un punto di ancoraggio.
OBJ_ARROW_LEFT_PRICE	Etichetta Prezzo a Sinistra	Un punto di ancoraggio.
OBJ_ARROW_RIGHT_PRICE	Etichetta Prezzo a Destra	Un punto di ancoraggio.
OBJ_ARROW_BUY	Segno di Buy	Un punto di ancoraggio.
OBJ_ARROW_SELL	Segno di Sell	Un punto di ancoraggio.
OBJ_ARROW	Freccia	Un punto di ancoraggio.
OBJ_TEXT	Testo	Un punto di ancoraggio.

ID	Descrizione	Punti di ancoraggio
OBJ_LABEL	Etichetta	La posizione viene impostata utilizzando la proprietà OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_BUTTON	Bottone	La posizione viene impostata utilizzando la proprietà OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_CHART	Grafico	La posizione viene impostata utilizzando la proprietà OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_BITMAP	Bitmap	Un punto di ancoraggio.
OBJ_BITMAP_LABEL	Etichetta Bitmap	La posizione viene impostata utilizzando la proprietà OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_EDIT	Modifica	La posizione viene impostata utilizzando la proprietà OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_EVENT	L'oggetto "Evento" corrispondente ad un evento nel calendario economico	Un punto di ancoraggio. Al momento, solo la coordinata temporale viene utilizzata.
OBJ_RECTANGLE_LABEL	L' oggetto "Etichetta Rettangolo" per la creazione e la progettazione dell'interfaccia grafica personalizzata.	La posizione viene impostata utilizzando la proprietà OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.

ObjectName

La funzione restituisce il nome dell'oggetto corrispondente nel grafico specificato, nella sottofinestra specificata, del tipo specificato.

```
string  ObjectName(
   long  chart_id,           // identificatore del grafico
   int   pos,                // numero in lista, degli oggetti
   int   sub_window=-1,      // indice finestra
   int   type=-1             // tipo di oggetto
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

pos

[in] numero ordinale dell'oggetto in base al filtro specificato dal numero e dal tipo della finestra secondaria.

sub_window=-1

[in] Numero di sottofinestra grafico. 0 significa la finestra grafico principale, -1 significa tutte le sotto-finestre del grafico, compresa la finestra principale.

type=-1

[in] Tipo dell' oggetto. Il valore può essere uno dei valori dell'enumerazione ENUM_OBJECT. -1 means all types.

Valore restituito

Il nome dell'oggetto viene restituito in caso di successo.

Nota

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Quando un oggetto viene rinominato, due eventi si formano simultaneamente. Questi eventi possono essere gestiti in un Expert Advisor o indicatore della funzione OnChartEvent():

- un evento di eliminazione di un oggetto con il vecchio nome;
- un evento di creazione di un oggetto con un nuovo nome.

ObjectDelete

La funzione rimuove l'oggetto con il nome specificato dalla tabella specificata.

```
bool  ObjectDelete(
   long    chart_id,      // identificatore del grafico
   string  name           // nome dell'oggetto
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[un] Nome dell' oggetto da eliminare.

Valore restituito

La funzione restituisce true se il comando è stato aggiunto con successo alla coda del chart specificato, altrimenti false.

Nota

Una chiamata asincrona viene sempre utilizzata per ObjectDelete(), che è ciò per cui la funzione restituisce solo i risultati di aggiunta comando ad una coda(queue) chart. In questo caso, true significa solo che il comando è stato accodato correttamente, ma il risultato della sua esecuzione non è noto.

Per verificare il risultato di esecuzione del comando, è possibile utilizzare la funzione ObjectFind() o qualsiasi altra funzione che richiede proprietà dell'oggetto, come ObjectGetXXX. Tuttavia, dovresti tenere presente che tali funzioni vengono aggiunte alla fine della coda di quel grafico ed aspettano il risultato dell'esecuzione (dovuto alla chiamata sincrona) e quindi possono richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Quando un oggetto viene rinominato, due eventi si formano simultaneamente. Questi eventi possono essere gestiti in un Expert Advisor o indicatore della funzione OnChartEvent():

- un evento di eliminazione di un oggetto con il vecchio nome;
- un evento di creazione di un oggetto con un nuovo nome.

ObjectsDeleteAll

Rimuove tutti gli oggetti dalla tabella specificata, sottofinestra grafico specificata, del tipo specificato.

```
int  ObjectsDeleteAll(
   long  chart_id,           // identificatore grafico
   int   sub_window=-1,      // indice finestra
   int   type=-1             // tipo di oggetto
   );
```

Rimuove tutti gli oggetti del tipo specificato utilizzando il prefisso nei nomi degli oggetti.

```
int  ObjectsDeleteAll(
   long           chart_id,  // chart ID
   const string     prefix,  // prefisso nel nome dell'oggetto
   int        sub_window=-1,  // indice finestra
   int        object_type=-1  // tipo d'oggetto
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

prefix

[in] Prefisso nei nomi degli oggetti. Tutti gli oggetti i cui nomi iniziano con questo insieme di caratteri verranno rimossi dal chart. Puoi specificare il prefisso come 'nome' o 'nome *' - entrambe le varianti funzioneranno lo stesso. Se viene specificata una stringa vuota come prefisso, gli oggetti con tutti i nomi possibili verranno rimossi.

sub_window=-1

[in] Numero di sottofinestra grafico. 0 significa la finestra grafico principale, -1 significa tutte le sotto-finestre del grafico, compresa la finestra principale.

type=-1

[in] Tipo dell' oggetto. Il valore può essere uno dei valori dell'enumerazione ENUM_OBJECT. -1 means all types.

Valore restituito

Restituisce il numero di oggetti eliminati. Per saperne di più sull' errore chiamare GetLastError().

Note

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

ObjectFind

La funzione cerca un oggetto con il nome specificato nella tabella con l'ID specificato.

```
int  ObjectFind(
longchart_id,// Identificatore del chart
   string  name          // nome dell'oggetto
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Il nome dell'oggetto ricercato.

Valore restituito

In caso di successo la funzione restituisce il numero della finestra secondaria (0 significa la finestra principale del grafico), in cui l'oggetto si trova. Se l'oggetto non viene trovato, la funzione restituisce un numero negativo. Per saperne di più sull' errore chiamare GetLastError().

Nota

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Quando un oggetto viene rinominato, due eventi si formano simultaneamente. Questi eventi possono essere gestiti in un Expert Advisor o indicatore della funzione OnChartEvent():

- un evento di eliminazione di un oggetto con il vecchio nome;
- un evento di creazione di un oggetto con un nuovo nome.

ObjectGetTimeByValue

La funzione restituisce il valore di tempo per il valore del prezzo specificato dell'oggetto specificato.

```
datetime  ObjectGetTimeByValue(
longchart_id,// Identificatore del chart
   string  name,          // nome dell'oggetto
   double  value,         // Prezzo
   int     line_id        // numero di riga
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

valore

[in] Valore prezzo.

line_id

[in] Identificatore linea.

Valore restituito

Il valore di tempo per il valore del prezzo specificato dell'oggetto specificato.

Nota

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Un oggetto può avere valori diversi in una coordinata prezzo, pertanto è necessario specificare il numero di riga. Questa funzione si applica solo per i seguenti oggetti:

- Trendline (OBJ_TREND)
- Trendline per angolo (OBJ_TRENDBYANGLE)
- Linea di Gann (OBJ_GANNLINE)
- Canale equidistante (OBJ_CHANNEL) - 2 lines
- Canale di regressione lineare (OBJ_REGRESSION) - 3 lines
- Canale di deviazione standard (OBJ_STDDEVCHANNEL) - 3 lines
- Linea con freccia (OBJ_ARROWED_LINE)

Vedi anche

Tipi di oggetti

ObjectGetValueByTime

La funzione restituisce il valore del prezzo per il valore di tempo specificato dell'oggetto specificato.

```
double  ObjectGetValueByTime(
   long      chart_id,     // identificatore del grafico
   string    name,         // nome dell'oggetto
   datetime  time,         // Tempo
   int       line_id       // numero di riga
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

time

[in] Valore temporale.

line_id

[in] ID Linea.

Valore restituito

Il valore del prezzo per il valore di tempo specificato dell'oggetto specificato.

Nota

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Un oggetto può avere valori diversi in una coordinata prezzo, pertanto è necessario specificare il numero di riga. Questa funzione si applica solo per i seguenti oggetti:

- Trendline (OBJ_TREND)
- Trendline per angolo (OBJ_TRENDBYANGLE)
- Linea di Gann (OBJ_GANNLINE)
- Canale equidistante (OBJ_CHANNEL) - 2 lines
- Canale di regressione lineare (OBJ_REGRESSION) - 3 lines
- Canale di deviazione standard (OBJ_STDDEVCHANNEL) - 3 lines
- Linea con freccia (OBJ_ARROWED_LINE)

Vedi anche

Tipi di oggetti

ObjectMove

La funzione cambia le coordinate del punto di ancoraggio specificato dell'oggetto.

```
bool  ObjectMove(
   long      chart_id,         // identificatore del grafico
   string    name,             // nome dell'oggetto
   int       point_index,      // numero del punto di ancoraggio
   datetime  time,             // Tempo
   double    price             // Prezzo
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

point_index

[in] Indice del punto di ancoraggio. Il numero di punti di ancoraggio dipende dal tipo di oggetto.

time

[in] Coordinate temporali del punto di ancoraggio selezionato.

price

[in] Prezzo coordinate del punto di ancoraggio selezionato.

Valore restituito

La funzione restituisce true se il comando è stato aggiunto con successo alla coda del chart specificato, altrimenti false.

Note

Una chiamata asincrona viene sempre utilizzata per ObjectMove(), che è ciò per cui la funzione restituisce solo i risultati di aggiunta comando ad una coda(queue) chart. In questo caso, true significa solo che il comando è stato accodato correttamente, ma il risultato della sua esecuzione non è noto.

Per verificare il risultato di esecuzione del comando, è possibile utilizzare una funzione che richiede le proprietà dell'oggetto, ad esempio ObjectGetXXX. Tuttavia, dovresti tenere presente che tali funzioni vengono aggiunte alla fine della coda di quel grafico ed aspettano il risultato dell'esecuzione (dovuto alla chiamata sincrona) e quindi possono richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

ObjectsTotal

La funzione restituisce il numero di oggetti nel grafico specificato, sottofinestra specificata, del tipo specificato.

```
int  ObjectsTotal(
   long  chart_id,           // identificatore del grafico
   int   sub_window=-1,      // indice finestra
   int   type=-1             // tipo di oggetto
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

sub_window=-1

[in] Numero di sottofinestra grafico. 0 significa la finestra grafico principale, -1 significa tutte le sotto-finestre del grafico, compresa la finestra principale.

type=-1

[in] Tipo dell' oggetto. Il valore può essere uno dei valori dell'enumerazione ENUM_OBJECT. -1 means all types.

Valore restituito

Il numero di oggetti.

Note

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

ObjectSetDouble

La funzione imposta il valore della proprietà dell'oggetto corrispondente. La proprietà dell' oggetto deve essere di tipo double. Ci sono due varianti della funzione.

Imposta il valore della proprietà, senza modificatore

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // proprietà
   double                        prop_value         // valore
   );
```

Imposta un valore di proprietà che indica il modificatore

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // proprietà
   int                           prop_modifier,     // modificatore
   double                        prop_value         // valore
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

prop_id

[in] ID della proprietà dell'oggetto. Il valore può essere uno dei valori dell'enumerazione ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Modificatore della proprietà specificata. Denota il numero del livello negli Strumenti di Fibonacci e nell'oggetto grafico Forche di Andrew's. La numerazione dei livelli parte da zero.

prop_value

[in] Il valore della proprietà.

Valore restituito

La funzione restituisce true solo se il comando per modificare le proprietà di un oggetto grafico sono state inviate con successo ad un grafico. In caso contrario, restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Note

La funzione utilizza una chiamata asincrona, il che significa che la funzione non attende l'esecuzione del comando che è stato aggiunto alla coda del grafico specificato. Invece, restituisce immediatamente il controllo.

Per verificare il risultato di esecuzione del comando, è possibile utilizzare una funzione che richiede la proprietà dell'oggetto specificato. Tuttavia, è necessario tenere presente che tali funzioni vengono aggiunte alla fine della coda di quel chart, ed aspettano il risultato dell'esecuzione, e quindi possono richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Esempio di creazione di un oggetto Fibonacci e l'aggiunta di un nuovo livello in esso

```
//+------------------------------------------------------------------+
//| Funzione di avvio del programma Script                           |
//+------------------------------------------------------------------+
voidOnStart()
  {
//--- array ausiliari
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- Copia i prezzi di apertura - le ultime 100 barre sono sufficienti
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("Fallimento nel tentativo di copiare i valori della serie prezzi High");
      return;
     }
//--- Copia il prezzo vicino - le ultime 100 barre sono sufficienti
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("Impossibile copiare i valori della serie prezzo High");
      return;
     }
//--- Copia il tempo di apertura per lei ultime 100 barre
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("Impossibile copiare i valori della serie prezzo Tempo");
      return;
     }
//--- Organizza l'accesso ai dati copiati come per le timeseries - indietro
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- Coordinate del primo punto di ancoraggio dell'oggetto Fibo
   price1=high[70];
   time1=time[70];
//--- Coordinate del secondo punto di ancoraggio dell'oggetto Fibo
```

```
   price2=low[50];
   time2=time[50];

//--- Tempo per creare l'oggetto Fibo
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
   if(created) // Se l'oggetto è stato creato con successo
     {
      //--- imposta i colori dei livelli Fibo
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- d'altro canto, quanti livelli di Fibo abbiamo?
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Fibo levels before = ",levels);
      //---output al Journal => numero di livellol:valore descrizione_livello
      for(int i=0;i<levels;i++)
        {
         Print(i,": ",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- Prova ad incrementare il numero di livelli per unità
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // fallimento nel cambiamento del numero di livelli
        {
         Print("Fallimento nel cambiamento del numero di livelli di Fibo, errore ",Get
        }
      //--- giusto per informare
      Print("Fibo levels after = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
      //--- imposta un valore per un nuovo livello creato
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // gestito per impostare il valore per un livello
        {
         Print("Successivamente imposta uno o più livelli Fibo");
         //--- Anche non dimenticare di impostare la descrizione del livello
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- Ottiene il valore attuale del numero dei livelli nell'oggetto Fibo
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("Livelli fibo dopo l'aggiunta = ",levels);
         //--- ancora una volta da in output tutti i livelli - giusto per assicurarsi
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }
        }
      else // Fallimento se si prova ad incrementare il numero dei livelli nell' ogget
        {
         Print("Fallimento ad impostare uno o più livelli Fibo. Error ",GetLastError()
        }
     }
```

```
}
```

Vedi anche

Object Types, Proprietà Oggetto

ObjectSetInteger

La funzione imposta il valore della proprietà dell'oggetto corrispondente. La proprietà oggetto deve essere di tipo datetime, int, colore, bool o char. Ci sono due varianti della funzione.

Imposta il valore della proprietà, senza modificatore

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificatore del grafico
   string                            name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // proprietà
   long                              prop_value         // value
   );
```

Imposta un valore di proprietà che indica il modificatore

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificatore del grafico
   string                            name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // proprietà
   int                               prop_modifier,     // modificatore
   long                              prop_value         // value
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

prop_id

[in] ID della proprietà dell'oggetto. Il valore può essere uno dei valori dell' enumerazione ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Modificatore della proprietà specificata. Esso denota il numero del livello negli strumenti di Fibonacci e nell'oggetto grafico Forche di Andrew's. La numerazione dei livelli parte da zero.

prop_value

[in] Il valore della proprietà.

Valore restituito

La funzione restituisce true solo se il comando per modificare le proprietà di un oggetto grafico sono state inviate con successo ad un grafico. In caso contrario, restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Note

La funzione utilizza una chiamata asincrona, il che significa che la funzione non attende l'esecuzione del comando che è stato aggiunto alla coda del grafico specificato. Invece, restituisce immediatamente il controllo.

Per verificare il risultato di esecuzione del comando, è possibile utilizzare una funzione che richiede la proprietà dell'oggetto specificato. Tuttavia, è necessario tenere presente che tali funzioni vengono aggiunte alla fine della coda di quel chart, ed aspettano il risultato dell'esecuzione, e quindi possono richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Un esempio di come creare una tabella di Colori Web

```
//+------------------------------------------------------------------------------------+
//|                                                          Tabella dei colori Web |
//|                                  Copyright 2011, MetaQuotes Software Corp |
//|                                                https://www.metaquotes.net |
//+------------------------------------------------------------------------------------+
#define X_SIZE 140      // spessore di un oggetto di modifica
#define Y_SIZE 33       // altezza di un oggetto di modifica
//+------------------------------------------------------------------------------------+
//| Array di colori web                                                                |
//+------------------------------------------------------------------------------------+
color ExtClr[140]=
  {
   clrAliceBlue,clrAntiqueWhite,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clrB
   clrBlue,clrBlueViolet,clrBrown,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate
   clrCornsilk,clrCrimson,clrCyan,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray
   clrDarkMagenta,clrDarkOliveGreen,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalm
   clrDarkSlateBlue,clrDarkSlateGray,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDee
   clrDodgerBlue,clrFireBrick,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,cl
   clrGoldenrod,clrGray,clrGreen,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,cl
   clrLavender,clrLavenderBlush,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCora
   clrLightGoldenrod,clrLightGreen,clrLightGray,clrLightPink,clrLightSalmon,clrLightSe
   clrLightSlateGray,clrLightSteelBlue,clrLightYellow,clrLime,clrLimeGreen,clrLinen,cl
   clrMediumAquamarine,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen
   clrMediumSpringGreen,clrMediumTurquoise,clrMediumVioletRed,clrMidnightBlue,clrMintC
   clrNavajoWhite,clrNavy,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrC
   clrPaleGreen,clrPaleTurquoise,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,c
   clrPurple,clrRed,clrRosyBrown,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,c
   clrSienna,clrSilver,clrSkyBlue,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,cl
   clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
  };
//+------------------------------------------------------------------------------------+
//| Creazione ed inizializzazione di un oggetto di modifica                            |
//+------------------------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- genera il nome per un nuovo oggetto di modifica
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- crea un nuovo oggetto di modifica
```

```
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("Non posso creare: '",name,"'");
      return;
     }
//--- imposta le coordinate, spessore ed altezza
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
   ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
//--- set text color
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- imposta colore di background
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- imposta il testo
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- crea la tabella 7x20 degli oggetti di modifica colorati
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

Vedi anche

Object Types, Proprietà Oggetto

ObjectSetString

La funzione imposta il valore della proprietà dell'oggetto corrispondente. La proprietà dell'oggetto deve essere di tipo string tipo. Ci sono due varianti della funzione.

Imposta il valore della proprietà, senza modificatore

```
bool  ObjectSetString(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // proprietà
   string                        prop_value         // valore
   );
```

Imposta un valore di proprietà che indica il modificatore

```
bool  ObjectSetString(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // proprietà
   int                           prop_modifier,     // modificatore
   string                        prop_value         // valore
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

prop_id

[in] ID della proprietà dell'oggetto. Il valore può essere uno dei valori dell' enumerazione ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Modificatore della proprietà specificata. Esso denota il numero del livello negli strumenti di Fibonacci e nell'oggetto grafico Forche di Andrew's. La numerazione dei livelli parte da zero.

prop_value

[in] Il valore della proprietà.

Valore restituito

La funzione restituisce true solo se il comando per modificare le proprietà di un oggetto grafico sono state inviate con successo ad un grafico. In caso contrario, restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Nota

La funzione utilizza una chiamata asincrona, il che significa che la funzione non attende l'esecuzione del comando che è stato aggiunto alla coda del grafico specificato. Invece, restituisce immediatamente il controllo.

Per verificare il risultato di esecuzione del comando, è possibile utilizzare una funzione che richiede la proprietà dell'oggetto specificato. Tuttavia, è necessario tenere presente che tali funzioni vengono aggiunte alla fine della coda di quel chart, ed aspettano il risultato dell'esecuzione, e quindi possono richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Quando un oggetto viene rinominato, due eventi si formano simultaneamente. Questi eventi possono essere gestiti in un Expert Advisor o indicatore della funzione OnChartEvent():

- un evento di eliminazione di un oggetto con il vecchio nome;
- un evento di creazione di un oggetto con un nuovo nome.

ObjectGetDouble

La funzione restituisce il valore della proprietà dell'oggetto corrispondente. La proprietà dell' oggetto deve essere di tipo double. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
double  ObjectGetDouble(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificatore proprietà
   int                           prop_modifier=0    // modificatore proprietà, se ri
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile passata per riferimento, dall'ultimo parametro.

```
bool  ObjectGetDouble(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificatore proprietà
   int                           prop_modifier,     // modificatore proprietà
   double&                       double_var         // qui si accetta il valore del
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

prop_id

[in] ID della proprietà dell'oggetto. Il valore può essere uno dei valori dell'enumerazione ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Modificatore della proprietà specificata. Per la prima variante, il valore del modificatore di default è uguale a 0. La maggior parte delle proprietà non richiedono un modificatore. Esso denota il numero del livello negli strumenti di Fibonacci e nell'oggetto grafico Forche di Andrew's. La numerazione dei livelli parte da zero.

double_var

[Out] Variabile del tipo double, che ha ricevuto il valore della proprietà richiesta.

Valore restituito

Valore del tipo double per la prima variante convocata.

Per la seconda variante la funzione restituisce true, se la proprietà è mantenuta ed il valore è stato inserito nella variabile double_var, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Note

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

ObjectGetInteger

La funzione restituisce il valore della proprietà dell'oggetto corrispondente. La proprietà oggetto deve essere di tipo datetime, int, colore, bool o char. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
long  ObjectGetInteger(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // identificatore proprietà
   int                           prop_modifier=0    // modificatore proprietà, se
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile passata per riferimento, dall'ultimo parametro.

```
bool  ObjectGetInteger(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // identificatore proprietà
   int                           prop_modifier,     // modificatore proprietà
   long&                         long_var           // qui si accetta il valore del
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

prop_id

[in] ID della proprietà dell'oggetto. Il valore può essere uno dei valori dell' enumerazione ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Modificatore della proprietà specificata. Per la prima variante, il valore del modificatore di default è uguale a 0. La maggior parte delle proprietà non richiedono un modificatore. Esso denota il numero del livello negli strumenti di Fibonacci e nell'oggetto grafico Forche di Andrew's. La numerazione dei livelli parte da zero.

long_var

[out] Variabile di tipo long che riceve il valore della proprietà richiesta.

Valore restituito

Il valore long per la prima variante di chiamata.

Per la seconda variante la funzione restituisce true, se la proprietà è mantenuta e il valore è stato inserito nella variabile long_var, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Note

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

ObjectGetString

La funzione restituisce il valore della proprietà dell'oggetto corrispondente. La proprietà dell'oggetto deve essere di tipo string tipo. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
string  ObjectGetString(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificatore proprietà
   int                           prop_modifier=0    // modificatore proprietà, se ri
   );
```

2. Restituisce true o false, a seconda del successo della funzione. In caso di successo, il valore della proprietà è posto in una variabile passata per riferimento, dall'ultimo parametro.

```
bool  ObjectGetString(
   long                          chart_id,          // identificatore del grafico
   string                        name,              // nome dell'oggetto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificatore proprietà
   int                           prop_modifier,     // modificatore proprietà
   string&                       string_var         // qui si accetta il valore del
   );
```

Parametri

chart_id

[in] Identificatore del Grafico. 0 significa il grafico corrente.

name

[in] Nome dell'oggetto.

prop_id

[in] ID della proprietà dell'oggetto. Il valore può essere uno dei valori dell' enumerazione ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Modificatore della proprietà specificata. Per la prima variante, il valore del modificatore di default è uguale a 0. La maggior parte delle proprietà non richiedono un modificatore. Esso denota il numero del livello negli strumenti di Fibonacci e nell'oggetto grafico Forche di Andrew's. La numerazione dei livelli parte da zero.

string_var

[out] Variabile di tipo stringa che riceve il valore delle caratteristiche richieste.

Valore restituito

Valore di stringa per la prima versione della chiamata.

Per la seconda versione della chiamata restituisce true, se la proprietà è mantenuta e il valore è stato inserito nella variabile string_var, altrimenti restituisce false. Per saperne di più sull' errore chiamare GetLastError().

Nota

La funzione utilizza una chiamata sincrona, il che significa che la funzione attende l'esecuzione di tutti i comandi che sono stati accodati per questo chart prima della sua chiamata, per cui questa funzione può richiedere molto tempo. Questa funzione deve essere presa in considerazione quando si lavora con un gran numero di oggetti su un chart.

Quando un oggetto viene rinominato, due eventi si formano simultaneamente. Questi eventi possono essere gestiti in un Expert Advisor o indicatore della funzione OnChartEvent():

- un evento di eliminazione di un oggetto con il vecchio nome;
- un evento di creazione di un oggetto con un nuovo nome.

TextSetFont

La funzione imposta il tipo di carattere per la visualizzazione del testo utilizzando metodi di disegno, e restituisce il risultato di tale operazione. Il carattere Arial con la dimensione -120 (12 pt) viene utilizzato per impostazione predefinita.

```
bool  TextSetFont(
   const string  name,            // nome del font o percorso al file di font sul disc
   int           size,            // grandeza del carattere
   uint          flags,           // combinazione di flags
   int           orientation=0    // angolo di inclinazione del testo
   );
```

Parametri

name

[in] Nome font nel sistema o il nome della risorsa che contiene il tipo di carattere o il percorso del file del font sul disco.

grandezza

[in] La dimensione del carattere che può essere impostata utilizzando i valori positivi e negativi. In caso di valori positivi, la dimensione di un testo visualizzato non dipende dalla impostazioni di grandezza caratteri del sistema operativo. In caso di valori negativi, il valore è impostato in decimi di punto e la grandezza del testo dipende dalle impostazioni del sistema operativo ("scala standard" o "su larga scala"). Vedere la nota di seguito per ulteriori informazioni sulle differenze tra le modalità.

flags

[in] Combinazione di flags che descrive lo stile del carattere.

orientamento

[in] L'inclinazione orizzontale di testo per l'asse X, l'unità di misura è di 0,1 gradi. Ciò significa che *orientation=450* sta per un'inclinazione pari a 45 gradi.

Valore restituito

Restituisce true se il carattere corrente è stato installato correttamente, in caso contrario false. Possibili errori di codice:

- ERR_INVALID_PARAMETER(4003) - *name* presenta NULL o "" (empty string),
- ERR_INTERNAL_ERROR(4001) - errore sistema operativo (per esempio, un tentativo di creare un font non-esistente).

Nota

Se "::" è usato nel nome del carattere, il tipo di carattere viene scaricato dalla risorsa EX5. Se il *nome* del font viene specificato con un' estensione, il tipo di carattere viene scaricato dal file, se il percorso inizia da "\" o "/", il file viene cercato relativamente alla MQL5 directory. In caso contrario, viene cercato relativamente al percorso del file che EX5 chiamato dalla funzione TextSetFont().

La grandezza del carattere viene impostata con valori positivi o negativi. Questo fatto definisce la dipendenza della dimensione del testo dalle impostazioni del sistema operativo (scala grandezza).

- Se la grandezza è specificata da un numero positivo, questa grandezza si trasforma in unità di misura fisica di un dispositivo (pixel) quando si cambia il tipo di carattere logico in uno fisico, e

questa grandezza corrisponde all'altezza dei glifi simbolo scelto tra i font disponibili . Questo caso non è raccomandato quando il testo visualizzato dalla funzione TextOut() e quelli visualizzati dall'oggetto grafico OBJ_LABEL ("Label") devono essere usati insieme sul chart.

- Se la grandezza è specificata da un numero negativo, questo numero dovrebbe essere impostato in decimi di punto logico (-350 è pari a 35 punti logici) ed è divisa per 10. Un valore ottenuto viene poi trasformato in unità di misura fisiche di un dispositivo (pixel) e corrisponde al valore assoluto dell'altezza di un simbolo scelto tra i font disponibili. Moltiplica la grandezza del carattere specificata nelle proprietà dell'oggetto da -10 per rendere la dimensione di un testo sullo schermo simile a quella nrll'oggetto OBJ_LABEL.

I flag possono essere usati come combinazione di flag di stile con uno dei flag che specificano la larghezza del carattere. I nomi dei flags sono di seguito.

Flags per specificare lo stile del carattere

Flag	Descrizione
FONT_ITALIC	Italic
FONT_UNDERLINE	Sottolineato
FONT_STRIKEOUT	Barrato

Flags per specificare la larghezza del carattere

Flag
FW_DONTCARE
FW_THIN
FW_EXTRALIGHT
FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL
FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD
FW_DEMIBOLD
FW_BOLD
FW_EXTRABOLD
FW_ULTRABOLD
FW_HEAVY
FW_BLACK

Vedi anche

Risorse, ResourceCreate(), ResourceSave(), TextOut()

TextOut

La funzione consente di visualizzare un testo in un array personalizzato (buffer) e restituisce il risultato di tale operazione. L'array è stato designato per creare la risorsa grafica.

```
bool  TextOut(
   const string       text,          // testo visualizzato
   int                x,             // coordinata X
   int                y,             // coordinata Y
   uint               anchor,        // tipo di ancora
   uint               &data[],       // output buffer
   uint               width,         // larghezza buffer in pixels
   uint               height,        // altezza buffer in pixels
   uint               color,         // colore testo
   ENUM_COLOR_FORMAT  color_format   // formato del colore per l'output
   );
```

Parametri

text

[in] Testo visualizzato che verrà scritto nel buffer. Solo un testo su una-riga viene visualizzato.

x

[in] Coordinata X del punto di ancoraggio del testo visualizzato.

y

[in] Coordinata Y del punto di ancoraggio del testo visualizzato.

ancora

[in] Il valore dei 9 predefiniti metodi di visualizzazione del luogo del punto di ancoramento del testo. Il valore viene impostato da una combinazione di due flags - flags di allineamento testo orizzontale e verticale. I nomi dei flags sono elencati nella Nota seguente.

data[]

[in] buffer, in cui il testo viene visualizzato. Il buffer viene utilizzato per creare la risorsa del grafico.

width

[in] Larghezza Buffer in pixel.

height

[in] Altezza Buffer in pixel.

color

[in] Colore testo.

color_format

[in] Il formato del colore viene impostato dal valore dell'enumerazione ENUM_COLOR_FORMAT.

Valore restituito

Restituisce true se ha successo, altrimenti false.

Nota

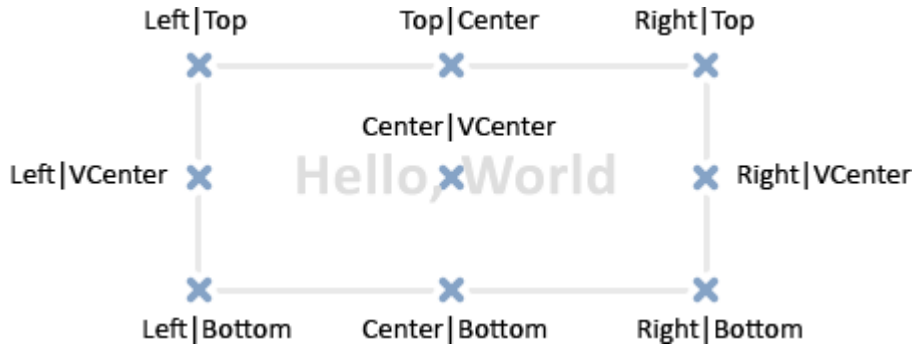
Il punto di ancoraggio specificato da *anchor* è una combinazione di due flag di allineamento testo orizzontale e verticale. Flags allineamento testo orizzontale:

- TA_LEFT - punto di ancoraggio sul lato sinistro del riquadro di delimitazione
- TA_CENTER - il punto di ancoraggio orizzontale si trova al centro del riquadro di delimitazione
- TA_RIGHT - punto di ancoraggio sul lato destro del riquadro di delimitazione

Flags allineamento testo verticale:

- TA_TOP - punto di ancoraggio sul lato superiore del riquadro di delimitazionee
- TA_VCENTER - punto di ancoraggio verticale si trova al centro del riquadro di delimitazione
- TA_BOTTOM - punto di ancoraggio sul lato inferiore del riquadro di delimitazione

Possibili combinazioni di flags e punti di ancoraggio specifici, sono mostrati nell'immagine.



Esempio:

```
//--- spessore ed altezza della tela (usata per il disegno)
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- mostra la finestra parametri prima di lanciare lo script
#property script_show_inputs
//--- abilitare per impostare il formato del colore
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- disegno array (buffer)
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------------------+
//| Funzione di avvio del programma Script                                       |
//+------------------------------------------------------------------------------+
voidOnStart()
  {
//--- crea l'oggetto OBJ_BITMAP_LABEL per il disegno
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- specifica il nome delle risorse grafiche per scrivere l'oggetto CLOCK
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- variabili ausiliari
   double a;            // angolo della freccia
   uint   nm=2700;      // angolo del minuto
   uint   nh=2700*12;   // angolo dell'ora
```

```
   uint   w,h;          // variabili per ricevere la grandezza della stringa testuale
   int    x,y;          // variabili per il calcolo delle coordinate correnti dei punt

//--- ruota le "lancette" dell'orologio in un loop infinito, finchè lo script viene in
   while(!IsStopped())
     {
      //--- ripulisce il buffer array del disegno dell'orologio
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- imposta il carattere per il disegno delle cifre per la facciata dell'orolo
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- disegna la facciata dell'orologio
      for(int i=1;i<=12;i++)
        {
         //--- riceve la grandezza dell'ora corrente sulla facciata dell'orologio
         TextGetSize(string(i),w,h);
         //--- calcola le coordinate dell'ora corrente sulla facciata dell'orologio
         a=-((i*300)%3600*M_PI)/1800.0;
         x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
         y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
         //--- da in output l'ora sulla facciata dell'orologio per il buffer ExtImg[]
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,c
        }
      //--- ora, specifica il carattere per il disegno della parte dei minuti
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- riceve la grandezza della parte dei minuti
      TextGetSize("----->",w,h);
      //--- calcola le coordinate della parte dei minuti sulla facciata dell'orologio
      a=-(nm%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- da in output la parte dei minuti sulla facciata dell'orologio nel buffer E
      TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_f

      //--- ora, imposta il font per il disegno della parte dei minuti
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- calcola le coordinata della parte dell'ora sulla facciata dell'orologio
      a=-(nh/12%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- da in output la parte dell'ora sulla facciata dell'orologio nel buffer Ext
      TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_forn

      //--- aggiorna la risorsa grafica
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //--- forza l'aggiornamento del chart
      ChartRedraw();

      //--- incrementa i contatori dell'ora e del minuto
      nm+=60;
```

```
      nh+=60;
      //--- mantiene una breve pausa tra i frames
      Sleep(10);
     }
//--- elimina l'oggetto CLOCK quando si completa l'operazione dello script
   ObjectDelete(0,"CLOCK");
//---
  }
```

Vedi anche

Risorse, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

TextGetSize

La funzione restituisce la larghezza della linea e l'altezza alle correnti impostazioni dei caratteri.

```
bool  TextGetSize(
   const string       text,          // stringa di testo
   uint&               width,        // larghezza del buffer in pixels
   uint&               height        // altezza del buffer in pixels
   );
```

Parametri

text

[in] Stringa, per cui lunghezza e larghezza devono essere ottenute.

width

[out] Parametro d'ingresso per la ricezione della larghezza.

height

[out] Parametro d'input per la ricezione di altezza.

Valore restituito

Restituisce true se ha successo, altrimenti false. Possibili errori di codice:

- ERR_INTERNAL_ERROR(4001) - errore sistema operativo.

Vedi anche

Risorse, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

Funzioni Indicatori Tecnici

Tutte le funzioni come iIMA iAC, iMACD, iIchimoku ecc creano una copia dell' indicatore tecnico corrispondente nella cache globale del terminale client. Se esiste già una copia dell'indicatore con tali parametri, la nuova copia non viene creata, ed il contatore dei riferimenti alle attuali copie aumenta.

Queste funzioni restituiscono l'handle della copia appropriata dell'indicatore. Inoltre, con questo handle, è possibile ricevere i dati calcolati dall' indicatore corrispondente. I corrispondenti dati del buffer (gli indicatori tecnici contengono dati calcolati nel loro buffer interni, che possono variare da 1 a 5, a seconda dell'indicatore) possono essere copiati su un programma-MQL5 utilizzando la funzione CopyBuffer().

Non si può fare riferimento ai dati indicatore giusto dopo che questo è stato creato, in quanto il calcolo dei valori degli indicatori richiede un po' di tempo, quindi è meglio creare gli handles in OnInit(). La funzione iCustom() crea l'indicatore personalizzato corrispondente, e restituisce il suo handle in caso venga creato con successo. Indicatori personalizzati possono contenere fino a 512 buffer indicatore, il cui contenuto può anche essere ottenuto con la funzione CopyBuffer(), utilizzando l' handle ottenuto.

Vi è un metodo universale per creare qualsiasi indicatore tecnico usando la funzione IndicatorCreate(). Questa funzione accetta i seguenti dati come parametri di input:

- nome del simbolo;
- timeframe;
- tipo dell'indicatore da creare;
- numero di parametri di input dell'indicatore;
- un array di tipo MqlParam contiene tutti i parametri di input necessari.

La memoria del computer può essere liberata da un indicatore che non è più utilizzato, usando la funzione IndicatorRelease(), per cui l'handle indicatore viene passato.

Note. Chiamate ripetuta della funzione dell'indicatore con gli stessi parametri in un programma-MQL5 non portano ad un aumento multiplo del contatore di riferimento; il contatore verrà aumentato solo una volta di 1. Tuttavia, si consiglia di ottenere gli handles degli indicatori nella funzione OnInit() o nel costruttore della classe, ed utilizzare ulteriormente tali handles in altre funzioni. Il contatore di riferimento diminuisce quando un programma-MQL5 viene deinizializzato.

Tutte le funzioni dell' indicatore hanno almeno 2 parametri - simbolo e periodo. Il valore NULL del simbolo significa il corrente simbolo, il valore 0 del periodo significa il corrente timeframe.

Funzione	Restituisce l'handle dell'indicatore:
iAC	Accelerator Oscillator
iAD	Accumulation/Distribution
iADX	Average Directional Index
iADXWilder	Average Directional Index by Welles Wilder
iAlligator	Alligator
iAMA	Adaptive Moving Average

Funzione	Restituisce l'handle dell'indicatore:
iAO	Awesome Oscillator
iATR	Average True Range
iBearsPower	Bears Power
iBands	Bollinger Bands®
iBullsPower	Bulls Power
iCCI	Commodity Channel Index
iChaikin	Chaikin Oscillator
iCustom	Custom indicator
iDEMA	Double Exponential Moving Average
iDeMarker	DeMarker
iEnvelopes	Envelopes
iForce	Force Index
iFractals	Fractals
iFrAMA	Fractal Adaptive Moving Average
iGator	Gator Oscillator
iIchimoku	Ichimoku Kinko Hyo
iBWMFI	Market Facilitation Index by Bill Williams
iMomentum	Momentum
iMFI	Money Flow Index
iMA	Moving Average
iOsMA	Moving Average of Oscillator (MACD histogram)
iMACD	Moving Averages Convergence-Divergence
iOBV	On Balance Volume
iSAR	Parabolic Stop And Reverse System
iRSI	Relative Strength Index
iRVI	Relative Vigor Index
iStdDev	Standard Deviation
iStochastic	Stochastic Oscillator
iTEMA	Triple Exponential Moving Average
iTriX	Triple Exponential Moving Averages Oscillator

Funzione	Restituisce l'handle dell'indicatore:
iWPR	Williams' Percent Range
iVIDyA	Variable Index Dynamic Average
iVolumes	Volumes

iAC

La funzione crea l' Accelerator Oscillator in una cache globale del terminale client e restituisce il suo handle. Ha un solo buffer.

```
int  iAC(
   string           symbol,     // nome simbolo
   ENUM_TIMEFRAMES  period      // periodo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori dell' enumerazione ENUM_TIMEFRAMES, 0 indica il corrente timeframe.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                              Demo_iAC.mq5 |
//|                                 Copyright 2011, MetaQuotes Software Corp. |
//|                                                          https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "del buffer indicatore per l'indicatore tecnico iAC."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- disegnare iAC
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,               // usa iAC
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iAC;          // tipo della funzione
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffers indicatore
double iACBuffer[];
double iACColors[];
//--- variabile per memorizzare l'handle dell'indicatore iAC
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell'indicatore Accelerator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
```

```
      {
       //--- dice riguardo il fallimento e l'output del codice di errore
       PrintFormat("Fallimento nel creare l'handle dell'indicatore iAC per il simbolo %
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- l'indicatore si ferma precocemente
       return(INIT_FAILED);
      }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Accelerator Oscilla
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iAC
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iACBuffer è maggiore del numero di valori dell'indicatore iAC
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

```
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempire gli array iACBuffer e iACColors con i valori dall'indicatore Accelerato
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Oscillator Accelerator
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempimento del buffer indicatore dall'indicatore iAC                        |
//+------------------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // buffer indicatore dei valori di
                          double &color_indexes[], // il color buffer (per memorizzare
                          int ind_handle,          // handle dell'indicatore iAC
                          int amount               // numero di valori copiati
                          )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell' array iACBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nella copia dei dati dall'indicatore iAC, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- ora copia gli indici dei colori
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i valori dall'indicatore iAC, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iAD

La funzione restituisce l'handle dell'indicatore Accumulation/Distribution. Ha un solo buffer.

```
int  iAD(
   string               symbol,             // nome del simbolo
   ENUM_TIMEFRAMES      period,             // periodo
   ENUM_APPLIED_VOLUME  applied_volume      // tipo di volume per il calcolo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori dell' enumerazione ENUM_TIMEFRAMES, 0 indica il corrente timeframe.

applied_volume

[in] Il volume usato. Può essere uno dei valori ENUM_APPLIED_VOLUME.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                              Demo_iAD.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "del buffer indicatore per l'indicatore tecnico iAD."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- traccio iAD
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,               // uso iAD
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iAD;          // tipo di funzione
input ENUM_APPLIED_VOLUME   volumes;                // volume usato
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffer indicatore
double          iADBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iAD
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell'indicatore Accumulazione/Distribuzione
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
```

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iAD per il simbolo %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe cui è calcolato l'indicatore Accumulation/Distributi
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall'indicatore iAD
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- se l' array iADBuffer è maggiore del numero di valori dell'indicatore iAD
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array iADBuffer con valori dell'indicatore Accumulation/Distribution
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizzare il numero di valori nell' indicatore Accumulation/Distribution
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iAD                                   |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // buffer indicatore della linea Accumula
                         int ind_handle,     // handle dell'indicatore iAD
                         int amount          // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte di iADBuffer con valori dal buffer indicatore che ha indice 0
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iAD, codice errore %d
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
```

```
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iADX

La funzione restituisce l'handle dell'indicatore Average Directional Movement Index.

```
int  iADX(
   string           symbol,          // nome simbolo
   ENUM_TIMEFRAMES  period,          // periodo
   int              adx_period       // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

adx_period

[in] Periodo per calcolare l'indice.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri buffer sono i seguenti: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                                  Demo_iADX.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                             https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "del buffer indicatore per l'indicatore tecnico iADX.
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

```
//--- traccio ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- traccio DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- disegna DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // usa iADX
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iADX;         // tipo della funzione
input int                   adx_period=14;          // periodo per il calcolo
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffers indicatore
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- variabile per conservare l'handle dell'indicatore iADX
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Average Directional Movement Inde
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
```

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iADX per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Average Di
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iADX
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iADXBuffer è maggiore del numero di valori dell'indicatore iAD
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array con i valori dell'indicatore Average Directional Movement Index
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori dell'indicatore Average Directional Movement Index
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iAC                             |
//+------------------------------------------------------------------------------+
```

```
bool FillArraysFromBuffers(double &adx_values[],      // buffer indicatore della linea
                           double &DIplus_values[],   // buffer indicatore per DI+
                           double &DIminus_values[],  // buffer indicatore for DI-
                           int ind_handle,            // handle dell'indicatore iADXW
                           int amount                 // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iADXBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iADX, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array DI_plusBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iADX, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array DI_plusBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iADX, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iADXWilder

La funzione restituisce l'handle di Index Average Directional Movement di Welles Wilder.

```
int  iADXWilder(
   string           symbol,          // nome simbolo
   ENUM_TIMEFRAMES  period,          // periodo
   int              adx_period       // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

adx_period

[in] Periodo per calcolare l'indice.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri buffer sono i seguenti: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          iADXWilder.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "del buffer indicatore per l'indicatore tecnico iADXWilder."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

```
//--- traccio ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- traccio DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- disegna DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // usa iADXWilder
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iADXWilder;   // tipo di funzione
input int                   adx_period=14;          // periodo per il calcolo
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffers indicatore
double          ADXBuffer[];
double          DI_plusBuffer[];
double          DI_minusBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iADXWilder
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Average Directional Movement Inde
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
```

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iADXWilder per il si
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Average Di
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iADXWilder
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iADXBuffer è maggiore del numero di valori dell'indicatore iAD
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array con i valori dell'indicatore Average Directional Movement Index
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori dell'indicatore Average Directional Movement Index
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iADXWilder                      |
//+------------------------------------------------------------------------------+
```

```
bool FillArraysFromBuffers(double &adx_values[],      // buffer indicatore della linea
                           double &DIplus_values[],   // buffer indicatore per DI+
                           double &DIminus_values[],  // buffer indicatore for DI-
                           int ind_handle,            // handle dell'indicatore iADXW
                           int amount                 // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iADXBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iADXWilder, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array DI_plusBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iADXWilder, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array DI_plusBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iADXWilder, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iAlligator

La funzione restituisce l'handle dell'indicatore Alligator.

```
int  iAlligator(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  jaw_period,        // periodo per il calcolo di jaws
   int                  jaw_shift,         // slittamento orizzontale di jaws(_*mascel
   int                  teeth_period,      // periodo per il calcolo di teeth
   int                  teeth_shift,       // slittamento orizzontale di teeth(_*denti
   int                  lips_period,       // periodo per il calcolo di lips
   int                  lips_shift,        // slittamento orizzontale di lips(_* labbr
   ENUM_MA_METHOD       ma_method,         // tipo di smussamento
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

jaw_period

[in] Periodo medio per la linea blu (Alligator's Jaw, _* mascella dell'Alligatore)

jaw_shift

[in] Lo shift per la linea blu relativa al grafico dei prezzi.

teeth_period

[in] Periodo medio per la linea rossa (Alligator's Teeth, _*denti dell'alligatore).

teeth_shift

[in] The shift of the red line relative to the price chart.

lips_period

[in] Periodo medio per la linea verde (labbra dell'Alligatore, _*Alligator's lips).

lips_shift

[in] Lo slittamento della linea verde rispetto al grafico dei prezzi.

ma_method

[in] Il metodo di calcolo della media. Può essere uno qualsiasi dei valori ENUM_MA_METHOD.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri del buffer sono i seguenti: 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                        Demo_iAlligator.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iAlligator."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili all' Alligator standard."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- traccio Jaws
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- traccio Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- traccio Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------------------+
```

```
//| Enumerazione dei metodi della creazione dell'handle                              |
//+----------------------------------------------------------------------------------+
enum Creation
  {
   Call_iAlligator,         // usa iAlligator
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iAlligator;   // tipo della funzione
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
input int                   jaw_period=13;          // periodo della linea Jaw
input int                   jaw_shift=8;            // slittamento della linea Jaw
input int                   teeth_period=8;         // periodo della linea Teeth
input int                   teeth_shift=5;          // slittamento della linea Teeth
input int                   lips_period=5;          // periodo della linea Lips
input int                   lips_shift=3;           // slittamento della linea Lips
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // medoto di media delle linee di A
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// tipo di prezzo utilizzato per
//--- buffers indicatore
double          JawsBuffer[];
double          TeethBuffer[];
double          LipsBuffer[];
//--- Variabile per memorizzare l'handle dell'indicatore iAlligator
int     handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Alligator
int     bars_calculated=0;
//+----------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                           |
//+----------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- impostiamo lo slittamento per ogni linea
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
```

```
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iAlligator)
      handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                        teeth_shift,lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[8];
     //--- periodi e slittamenti delle linee di Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
//--- tipo di smussamento
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- tipo di prezzo
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- crea l'handle
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iAlligator per il si
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Alligator
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(pe
```

```
                              jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iAlligator
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array JawsBuffer è maggiore del numero di valori dell'indicatore iAl
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore Alligator
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,l
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

```
                              short_name,
                              values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero dei valori nell'indicatore Alligator
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie i buffer dell' indicatore dall'indicatore iAlligator                       |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // buffer indicatore per la linea
                           int j_shift,            // slittamento per la linea Jaw
                           double &teeth_buffer[], // buffer per la linea Teeth
                           int t_shift,            // slittamento per la linea di Teet
                           double &lips_buffer[],  // buffer indicatore per la linea
                           int l_shift,            // slittamento per la linea di Lips
                           int ind_handle,         // handle dell'indicatore iAlligato
                           int amount              // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array JawsBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iAlligator, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array TeethBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iAlligator, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array LipsBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iAlligator, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
```

```
      }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iAMA

La funzione restituisce l'handle dell'indicatore Adaptive Moving Average. Ha un solo buffer.

```
int  iAMA(
   string               symbol,              // nome simbolo
   ENUM_TIMEFRAMES      period,              // periodo
   int                  ama_period,          // periodo medio per AMA
   int                  fast_ma_period,      // periodo veloce MA
   int                  slow_ma_period,      // periodo lento MA
   int                  ama_shift,           // slittamento orizzontale dell'indicatore
   ENUM_APPLIED_PRICE   applied_price        // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ama_period

[in] Il periodo di calcolo, in cui il coefficiente di efficienza viene calcolato.

fast_ma_period

[in] Il periodo veloce per il calcolo coefficiente di smussamento per un mercato rapido.

slow_ma_period

[in] Il periodo lento per il calcolo coefficiente di smussamento per un mercato rapido.

ama_shift

[in] Slittamento dell'indicatore rispetto al grafico dei prezzi.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                                 Demo_iAMA.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                                 https://www.mql5.com |
//+------------------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iAMA."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili all' AMA standard."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- traccio iAMA
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // uso iAMA
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iAMA;          // tipo di funzione
input string                symbol=" ";              // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // timeframe
input int                   ama_period=15;           // periodo per il calcolo
input int                   fast_ma_period=2;        // periodo di MA veloce
input int                   slow_ma_period=30;       // periodo di MA lento
input int                   ama_shift=0;             // slittamento orizzontale
input ENUM_APPLIED_PRICE    applied_price;           // tipo di prezzo
//--- buffer indicatore
double         iAMABuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iAMA
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
```

```
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- imposta lo slittamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[5];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ama_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=fast_ma_period;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slow_ma_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ama_shift;
      //--- tipo di prezzo
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iAMA per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Adaptive Moving Ave
```

```
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iAMA
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iAMABuffer è maggiore del numero di valori dell'indicatore iAM
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore Adaptive Moving Average
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

```
                              short_name,
                              values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Adaptive Moving Average
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iAMA                                  |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // buffer indicatore per la linea di A
                         int a_shift,           // slittamento per la linea di AMA
                         int ind_handle,        // handle per la linea dell'indicatore
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iAMABuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iAMA, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iAO

La funzione restituisce l'handle dell'indicatore Awesome oscillator. Ha un solo buffer.

```
int  iAO(
   string           symbol,     // nome simbolo
   ENUM_TIMEFRAMES  period      // periodo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------------+
//|                                                                        Demo_iAO.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                                                https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iAO."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- il tracciamento di iAO
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
```

```
//| Enumerazione dei metodi della creazione dell'handle                              |
//+----------------------------------------------------------------------------------+
enum Creation
  {
   Call_iAO,               // usa iAO
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iAO;          // tipo della funzione
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffers indicatore
double          iAOBuffer[];
double          iAOColors[];
//--- variabile per memorizzare l'handle dell'indicatore iAO
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell'indicatore Awesome Oscillator
int    bars_calculated=0;
//+----------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                           |
//+----------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
```

```
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iAO per il simbolo %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Awesome Oscillator
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iAO
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iADBuffer è maggiore del numero di valori dell'indicatore iAO
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
```

```
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempire gli array iAOBuffer e iAOColors con i valori dall'indicatore Awesome Os
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Oscillator Accelerator
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempimento del buffer indicatore dall'indicatore iAO                              |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // buffer indicatore dei valori di
                          double &color_indexes[], // il color buffer (per memorizzare
                          int ind_handle,          // handle of the iAO indicator
                          int amount               // numero di valori copiati
                          )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iAOBuffer con valori dal buffer indicatore che ha i
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iAO, codice errore %d
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- ora copia gli indici dei colori
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Failed to copy color values from the iAO indicator, error code %d",
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
```

```
//| Funzione deinizializzazione indicatore                                   |
//+------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iATR

La funzione restituisce l'handle della indicatore Average True Range. Ha un solo buffer.

```
int  iATR(
   string           symbol,        // nome simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Il valore del periodo medio per il calcolo dell'indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iATR.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iATR."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- traccio iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// uso iATR
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input int                   atr_period=14;          // periodo per il calcolo
input Creation              type=Call_iATR;          // tipo della funzione
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffer indicatore
double          iATRBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iAC
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Average True Range
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
```

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iATR per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Average Tru
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iATR
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- se l' array iATRBuffer è maggiore del numero di valori dell'indicatore iA
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array con i valori dell'indicatore Average True Range
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è an
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Oscillator Accelerator
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iATR                                  |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di ATR
                         int ind_handle,    // handle dell'indicatore iATR
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iATRBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iATR, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iBearsPower

La funzione restituisce l'handle dell'indicatore Bears Power. Ha un solo buffer.

```
int  iBearsPower(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Il valore del periodo medio per il calcolo dell'indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                      Demo_iBearsPower.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iBearsPower."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iBearsPower
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // uso iBearsPower
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iBearsPower;  // tipo della funzione
input int                   ma_period=13;           // periodo della media mobile
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffer indicatore
double          iBearsPowerBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iBearsPower
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell'indicatore Bears Power
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
```

```
      MqlParam pars[1];
      //--- period of ma
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iBearsPower per il s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Bears Power
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iBearsPower
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

```
      {
       //--- se l' array iBearsPowerBuffer è maggiore del numero di valori dell'indicat
       //--- altrimenti, copiamo meno della grandezza del buffer indicatore
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
       //--- per il calcolo non viene aggiunta più di una barra
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- riempie gli array iBearsPowerBuffer con valori dell'indicatore Bears Power
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
    if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
    string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
    Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Bears Power
    bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
    return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iBearsPower                     |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Bears P
                         int ind_handle,    // handle dell'indicatore iBearsPower
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
    ResetLastError();
//--- riempie una parte dell'array iBearsPowerBuffer con valori dal buffer indicatore
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- se la copia fallisce, dice il codice dell'errore
       PrintFormat("Fallimento nel copiare i dati dall'indicatore iBearsPower, codice e
       //--- esce con risultato zero - significa che l'indicatore è considerato come no
       return(false);
      }
//--- tutto è ok
    return(true);
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione deinizializzazione indicatore                                   |
//+------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iBands

La funzione restituisce l'handle dell'indicatore Bollinger Bands®.

```
int  iBands(
   string               symbol,           // nome del simbolo
   ENUM_TIMEFRAMES      period,           // periodo
   int                  bands_period,     // periodo per il calcolo medio della linea
   int                  bands_shift,      // slittamento orizzontale dell'indicatore
   double               deviation,        // numero di deviazioni standard
   ENUM_APPLIED_PRICE   applied_price     // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

bands_period

[in] Il periodo medio della linea principale dell'indicatore.

bands_shift

[in] Lo slittamento dell'indicatore relativo al grafico dei prezzi.

deviazione

[in] Deviazione dalla linea principale.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri del buffer sono i seguenti: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iBands.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                             https://www.mql5.com |
//+------------------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iBands."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- il tracciamento di Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- il tracciamento Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- il tracciamento Middle
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,             // uso iBands
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation             type=Call_iBands;          // tipo di funzione
input int                  bands_period=20;           // periodo della media mobile (
input int                  bands_shift=0;             // slittamento
input double               deviation=2.0;             // numero di deviazioni standard
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo di prezzo
input string               symbol=" ";                // simbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffers indicatore
double         UpperBuffer[];
double         LowerBuffer[];
```

```
double         MiddleBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iBands
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell'indicatore Bollinger Bands
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- impostiamo lo slittamento per ogni linea
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[4];
      //--- periodo di ma
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- slittamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- numero di deviazione standard
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
```

```
      //--- tipo di prezzo
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iBands per il simbol
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Bollinger Bands
   short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
                           bands_period,bands_shift,deviation,EnumToString(applied_pri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iBands
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

```
     {
      //--- se la grandezza dei buffer indicatore è maggiore del numero di valori dell
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore Bollinger Bands
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Bollinger Bands
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iBands                          |
//+------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // buffer indicatore della linea
                           double &upper_values[],    // buffer indicatore del bordo s
                           double &lower_values[],    // buffer indicatore del bordo i
                           int shift,                 // slittamento
                           int ind_handle,            // handle dell'indicatore iBands
                           int amount                 // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array MiddleBuffer con valori dal buffer indicatore che h
   if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iBands, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
```

```
//--- riempie una parte dell'array UpperBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iBands, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array LowerBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iBands, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iBullsPower

La funzione restituisce l'handle dell' indicatore Bulls Power. Ha un solo buffer.

```
int  iBullsPower(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Il periodo medio per il calcolo dell'indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                      Demo_iBullsPower.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iBullsPower."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iBullsPower
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // uso iBullsPower
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iBullsPower;  // tipo della funzione
input int                   ma_period=13;           // periodo della media mobile
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffer indicatore
double          iBullsPowerBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iBullsPower
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell'indicatore Bulls Power
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
```

```
      MqlParam pars[1];
      //--- periodo di ma
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iBullsPower per il s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Bulls Power
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iBullsPower
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

```
      {
       //--- se l' array iBullsPowerBuffer è maggiore del numero di valori dell'indicat
       //--- altrimenti, copiamo meno della grandezza del buffer indicatore
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
       //--- per il calcolo non viene aggiunta più di una barra
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- riempie gli array iBullsPowerBuffer con valori dell'indicatore Bulls Power
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
    if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
    string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
    Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Bulls Power
    bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
    return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iBullsPower                     |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Bulls P
                         int ind_handle,    // handle dell'indicatore iBullsPower
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
    ResetLastError();
//--- riempie una parte dell'array iBullsPowerBuffer con valori dal buffer indicatore
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- se la copia fallisce, dice il codice dell'errore
       PrintFormat("Fallimento nel copiare i dati dall'indicatore iBullsPowerBuffer, co
       //--- esce con risultato zero - significa che l'indicatore è considerato come no
       return(false);
      }
//--- tutto è ok
    return(true);
  }
//+------------------------------------------------------------------------------+
```

```
//| Funzione deinizializzazione indicatore                                   |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
//+------------------------------------------------------------------------------+
```

iCCI

La funzione restituisce l'handle dell'indicatore Commodity Index Channel. Ha un solo buffer.

```
int  iCCI(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Il periodo medio per il calcolo dell'indicatore.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                              Demo_iCCI.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iCCI."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- il tracciamento di iCCI
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- livelli orizzontali nella finestra dell'indicatore
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // usa iCCI
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iCCI;               // tipo della funzione
input int                   ma_period=14;                 // periodo di media mobile
input ENUM_APPLIED_PRICE    applied_price=PRICE_TYPICAL;  // tipo di prezzo
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffer indicatore
double         iCCIBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iCCI
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Commodity Channel Index
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
```

```
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[2];
      //--- periodo della media mobile
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo di prezzo
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iCCI per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Bollinger Bands®
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

```
  {
//--- numero di valori copiati dall' indicatore iCCI
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iCCIBuffer è maggiore del numero di valori dell'indicatore iCC
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array iCCIBuffer con i valori dell'indicatore Commodity Channel Index
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizziamo il numero di valori dell'indicatore Commodity Channel Index
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iCCI                            |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Commodi
                         int ind_handle,    // handle dell'indicatore iCCI
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
```

```
   ResetLastError();
//--- riempie una parte dell'array iCCIBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iCCI, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iChaikin

La funzione restituisce l'handle dell'indicatore Chaikin Oscillator. Ha un solo buffer.

```
int  iChaikin(
   string               symbol,             // nome del simbolo
   ENUM_TIMEFRAMES      period,             // periodo
   int                  fast_ma_period,     // periodo fast
   int                  slow_ma_period,     // periodo slow
   ENUM_MA_METHOD       ma_method,          // tipo di smussamento
   ENUM_APPLIED_VOLUME  applied_volume      // tipo di volume
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

fast_ma_period

[in] Periodo medio di Fast per i calcoli.

slow_ma_period

[in] Periodo medio di Slow per i calcoli.

ma_method

[in] Tipo di smussamento. Può essere una delle costanti di media ENUM_MA_METHOD.

applied_volume

[in] Il volume utilizzato. Può essere una delle costanti di ENUM_APPLIED_VOLUME.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_iChaikin.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
```

```
#property description "dei buffers indicatore per l'indicatore tecnico iChaikin."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iChaikin plot
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,          // uso iChaikin
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iChaikin;           // tipo di funzione
input int                   fast_ma_period=3;             // periodo di fast ma
input int                   slow_ma_period=10;            // periodo di slow ma
input ENUM_MA_METHOD        ma_method=MODE_EMA;           // tipo di smussamento
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo di volume
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffer indicatore
double         iChaikinBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iChaikin
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Chaikin Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
```

```
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[4];
      //--- periodo di fast ma
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- periodo di slow ma
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- tipo di smussamento
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo di volume
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iChaikin per il simb
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Chaikin Oscillator
   short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period)
                           fast_ma_period,slow_ma_period,
                           EnumToString(ma_method),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
```

```
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iChaikin
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iChaikinBuffer è maggiore del numero di valori dell'indicatore
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array iChaikinBuffer con valori dell'indicatore Chaikin Oscillator
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori dell'indicatore Chaikin Oscillator
```

```
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iChaikin |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Chaikin
                         int ind_handle,    // handle dell'indicatore iChaikin
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iChaikinBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iChaikin, codice erro
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iCustom

La funzione restituisce l'handle di un indicatore personalizzato specificato.

```
int  iCustom(
   string           symbol,     // nome simbolo
   ENUM_TIMEFRAMES  period,     // periodo
   string           name        // cartella/nome_indicatore_personalizzato
   ...                          // elenco dei parametri di unput dell'indicatore
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

name

[in] Nome dell'indicatore personalizzato (custom indicator). Se il nome inizia con la barra inversa '\', il file dell'indicatore EX5 viene ricercato rispetto alla directory principale dell'indicatore MQL5\Indicators. Pertanto, quando si chiama iCustom(Simbol(), Period(), "\FirstIndicator "...), l'indicatore viene scaricato come MQL5\Indicators\FirstIndicator.ex5. Se il percorso non contiene file, si verifica l'errore 4802 (ERR_INDICATOR_CANNOT_CREATE).

Se il percorso non inizia con '\', l'indicatore viene cercato e scaricato come segue:

- Innanzitutto, il file indicatore EX5 viene cercato nella cartella in cui si trova il file EX5 del programma chiamante. Ad esempio, CrossMA.EX5 EA si trova in MQL5\Experts\MyExperts e contiene la chiamata iCustom (Symbol(), Period(), "SecondIndicator" ...). In questo caso, l'indicatore viene cercato in MQL5\Experts\MyExperts\SecondIndicator.ex5.
- Se l'indicatore non viene trovato nella stessa directory, la ricerca viene eseguita in relazione alla directory principale dell'indicatore MQL5\Indicators. In altre parole, viene eseguita la ricerca del file MQL5\Indicators\SecondIndicator.ex5. Se l'indicatore non viene trovato ancora, la funzione restituisce INVALID_HANDLE e viene generato l'errore 4802 (ERR_INDICATOR_CANNOT_CREATE).

Se il percorso dell'indicatore è impostato nella sottodirectory (ad esempio MyIndicators\ThirdIndicator), la ricerca viene prima eseguita nella cartella del programma chiamante (l'EA si trova in MQL5\Experts\MyExperts) in MQL5\Experts\MyExperts\MyIndicators\ThirdIndicator.ex5. In caso di esito negativo, viene eseguita la ricerca del file MQL5\Indicators\MyIndicators\ThirdIndicator.ex5. Assicurati di usare la doppia barra inversa '\\' come separatore nel percorso, ad esempio iCustom (Symbol(), Period(), "MyIndicators\\ThirdIndicator"...)

...

[in]parametri-di-input- di un indicatore personalizzato, separati da virgole. Tipo ed ordine con cui i parametri devono corrispondere. Se non ci sono i parametri specificati, allora verranno utilizzati valori predefiniti.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

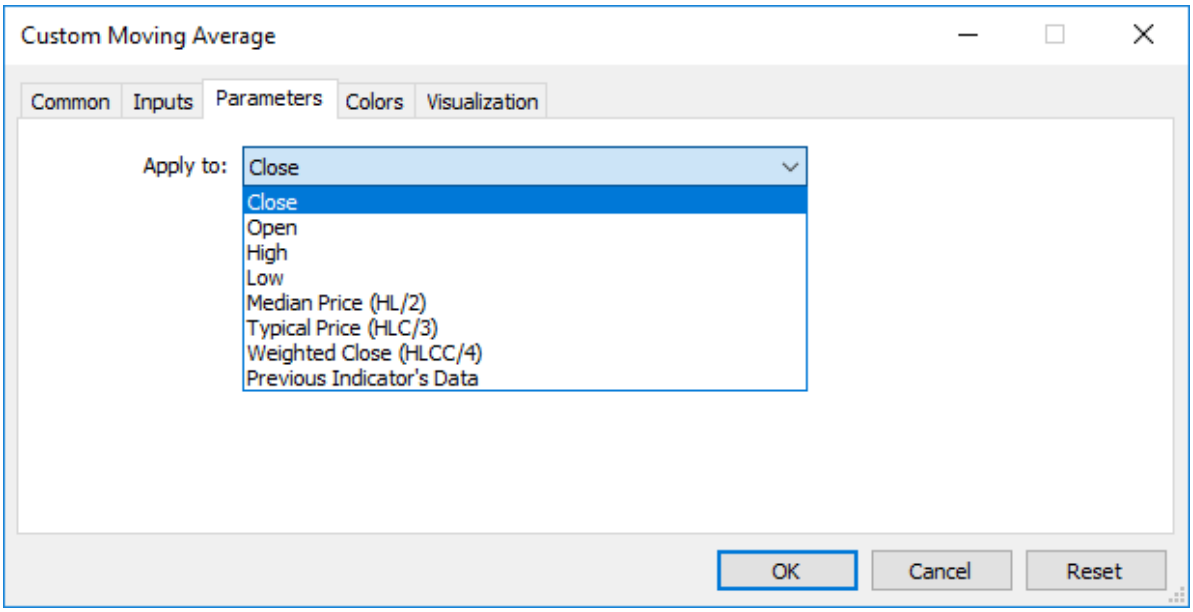
Nota

Un indicatore personalizzato deve essere compilato (con estensione EX5) e si trova nella directory MQL5/Indicators del terminale client o la sua sottodirectory.

Indicatori che richiedono il testing vengono definiti automaticamente dalla chiamata della funzione iCustom(), se il parametro corrispondente è impostato attraverso la costante stringa. Per tutti gli altri casi (utilizzo della funzione IndicatorCreate() o uso di una stringa non-costante nel parametro che imposta il nome indicatore) la proprietà #property tester_indicator è necessaria:

```
#property tester_indicator "indicator_name.ex5"
```

Se la prima forma di chiamata viene utilizzata nell'indicatore, allora all'avvio dell' indicatore personalizzato è possibile inoltre indicare i dati per il calcolo nella sua scheda "Parametri". Se il parametro "Applica a" non è selezionato in modo esplicito, il calcolo predefinito è basato sui valori dei prezzi "Close".



Quando si chiama un indicatore personalizzato da un programma MQL5, il parametro Applied_Price o un handle di un altro indicatore dovrebbe essere passato per ultimo, dopo tutte le variabili di input dell'indicatore personalizzato.

Vedi anche

Proprietà del programma, TimeSeries ed Accesso Indicatori,IndicatorCreate(), IndicatorRelease()

Esempio:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
```

```
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parametri di input
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- buffers indicatore
double         Label1Buffer[];
//--- Handle dell'indicatore personalizzato Custom Moving Average.mq5
int MA_handle;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- mappatura buffers indicatore
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // usando i prezzi close
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- Copiare i valori dell'indicatore Custom Moving Average nel nostro buffer di ind
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- Se il nostro tentativo è fallito - Segnala questo
   if(copy<=0)
      Print("Il tentativo di ottenere i valori di Custom Moving Average è fallito");
```

```
//--- restituisce il valore di prev_calculated per la prossima chiamata
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
```

iDEMA

La funzione restituisce l'handle dell'indicatore Double Exponential Moving Average. Ha un solo buffer.

```
int  iDEMA(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   int                  ma_shift,          // slittamento orizzontale
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio (conteggio barre) per i calcoli.

ma_shift

[in] Slittamento dell'indicatore rispetto al grafico dei prezzi.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iDEMA.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iDEMA."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
```

```
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iDEMA
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,              // uso iDEMA
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iDEMA;           // tipo della funzione
input int                   ma_period=14;              // periodo della media mobile
input int                   ma_shift=0;                // slittamento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double          iDEMABuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iDEMA
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Double Exponential Moving Averag
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //--- set shift
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
```

```
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[3];
      //--- periodo della media mobile
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- slittamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo di prezzo
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iDEMA per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Double Exponential
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iDEMA
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iDEMABuffer è maggiore del numero di valori dell'indicatore iD
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array iDEMABuffer con valori dell'indicatore Double Exponential Movin
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Double Exponential Moving Average
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iDEMA                           |
```

```
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Double
                         int shift,         // slittamento
                         int ind_handle,    // handle dell'indicatore iDEMA
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iDEMABuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iDEMA, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iDeMarker

La funzione restituisce l'handle dell'indicatore DeMarker. Ha un solo buffer.

```
int  iDeMarker(
   string           symbol,        // nome simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio (conteggio barre) per i calcoli.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                        Demo_iDeMarker.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iDeMarker."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iDeMarker
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- livelli orizzontali nella finestra dell'indicatore
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,          // uso iDeMarker
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iDeMarker;       // tipo della funzione
input int                   ma_period=14;              // periodo della media mobile
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double          iDeMarkerBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iDeMarker
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore DeMarker
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
```

```
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[1];
      //--- periodo della media mobile
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iDeMarker per il sir
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore DeMarker
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iDeMarker
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
```

```
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array v è maggiore del numero di valori dell'indicatore iDeMarker pe
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array iDeMarkerBuffer con valori dell'indicatore DeMarker
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero dei valori nell'indicatore DeMarker
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iDeMarker                             |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di DeMarke
                         int ind_handle,    // handle dell'indicatore iDeMarker
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iDeMarkerBuffer con valori dal buffer indicatore ch
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iDeMarker, codice err
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iEnvelopes

La funzione restituisce l'handle dell'indicatore Envelopes.

```
int  iEnvelopes(
   string               symbol,           // nome del simbolo
   ENUM_TIMEFRAMES      period,           // periodo
   int                  ma_period,        // periodo per il calcolo della linea media
   int                  ma_shift,         // slittamento orizzontale dell'indicatore
   ENUM_MA_METHOD       ma_method,        // tipo di smussamento
   ENUM_APPLIED_PRICE   applied_price,    // tipo di prezzo o handle
   double               deviation         // deviazione dei confini dalla linea media
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio per la linea principale.

ma_shift

[in] Lo slittamento del relativo indicatore al grafico dei prezzi.

ma_method

[in] Tipo di smussamento. Può essere uno dei valori di ENUM_MA_METHOD.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

deviazione

[in] La deviazione dalla linea principale (in percentuale).

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri di buffer: 0 - UPPER_LINE, 1 - LOWER_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
```

```
//|                                                                Demo_iEnvelopes.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                                https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iEnvelopes."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- il tracciamento di Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- il tracciamento Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,         // uso iEnvelopes
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iEnvelopes;      // tipo della funzione
input int                   ma_period=14;              // periodo della media mobile
input int                   ma_shift=0;                // slittamento
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // tipo di smussamento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input double                deviation=0.1;             // deviazione dei bordi dalla me
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double         UpperBuffer[];
double         LowerBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iEnvelopes
```

```
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Envelopes
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- impostiamo lo slittamento per ogni linea
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviati
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[5];
      //--- periodo della media mobile
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- slittamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo di smussamento
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo di prezzo
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- tipo di prezzo
```

```
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iEnvelopes per il si
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Envelopes
   short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(pe
   ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iEnvelopes
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- se l' array UpperBuffer è maggiore del numero di valori dell'indicatore iE
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempire gli array UpperBuffer e LowerBuffer con i valori dall'indicatore Envelo
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero dei valori nell'indicatore Envelopes
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iEnvelopes                            |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],    // buffer indicatore per il bord
                           double &lower_values[],    // indicatore del bordo inferio
                           int shift,                 // slittamento
                           int ind_handle,            // handle dell'indicatore iEnvel
                           int amount                 // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array UpperBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iEnvelopes, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- riempie una parte dell'array LowerBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
```

```
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iEnvelopes, codice er
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iForce

La funzione restituisce l'handle dell'indicatore Force Index. Ha un solo buffer.

```
int  iForce(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   ENUM_MA_METHOD       ma_method,         // tipo di smussamento
   ENUM_APPLIED_VOLUME applied_volume      // tipo di volume per il calcolo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio per il calcolo degli indicatori.

ma_method

[in] Tipo di smussamento. Può essere uno dei valori di ENUM_MA_METHOD.

applied_volume

[in] Il volume utilizzato. Può essere uno dei valori di ENUM_APPLIED_VOLUME.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_iForce.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iForce."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- tracciamento iForce
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,             // uso iForce
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iForce;             // tipo della funzione
input int                   ma_period=13;                 // periodo di media
input ENUM_MA_METHOD        ma_method=MODE_SMA;           // tipo di smussamento
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo di volume
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffer indicatore
double          iForceBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iForce
int     handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Force
int     bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
```

```
      {
       //--- prende il simbolo del grafico indicatore a cui è attaccato
       name=_Symbol;
      }
//--- crea l'handle dell'indicatore
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[3];
      //--- periodo della media mobile
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo di smussamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- tipo di volume
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- tipo di prezzo
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iForce per il simbol
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Force
   short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                           ma_period,EnumToString(ma_method),EnumToString(applied_volu
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iForce
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iForceBuffer è maggiore del numero di valori dell'indicatore i
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array iForceBuffer con valori dell'indicatore Force
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero dei valori nell'indicatore Force
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iForce                          |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Force
```

```
                          int ind_handle,     // handle dell'indicatore iForce
                          int amount          // numero di valori copiati
                          )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iForceBuffer con valori dal buffer indicatore che h
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iForce, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iFractals

La funzione restituisce l'handle dell'indicatore Fractals.

```
int  iFractals(
   string           symbol,     // nome simbolo
   ENUM_TIMEFRAMES  period      // periodo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri del buffer sono i seguenti: 0 - UPPER_LINE, 1 - LOWER_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_iFractals.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iFractals."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- il tracciamento di FractalUp
```

```
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- il tracciamento di FractalDown
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,          // uso iFractals
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iFractals;          // tipo della funzione
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffers indicatore
double          FractalUpBuffer[];
double          FractalDownBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iFractals
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Fractals
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- imposta i codici usando un simbolo dal set di caratteri Wingdings per la proprie
   PlotIndexSetInteger(0,PLOT_ARROW,217); // freccia su
   PlotIndexSetInteger(1,PLOT_ARROW,218); // freccia giu
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
```

```
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iFractals per il sir
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Fractals
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iFractals
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'
```

```
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array FractalUpBuffer è maggiore del numero di valori dell'indicato
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempire gli array FractalUpBuffer e FractalDownBuffer con i valori dall'indicat
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero dei valori nell'indicatore Fractals
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iFractals                             |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // buffer indicatore per la fre
                           double &down_arrows[],      // buffer indicatore per la fre
                           int ind_handle,             // handle dell'indicatore iFrac
                           int amount                  // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iBullsPowerBuffer con valori dal buffer indicatore
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iFractals nell'array
                  GetLastError());
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
```

```
//--- riempie una parte dell'array FractalDownBuffer con valori dal buffer indicatore
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iFractals nell'array
                  GetLastError());
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iFrAMA

La funzione restituisce l'handle dell'indicatore Fractal Adaptive Moving Average. Ha un solo buffer.

```
int  iFrAMA(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   int                  ma_shift,          // slittamento orizzontale del grafico
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo (conteggio barre) per i calcoli dell'indicatore.

ma_shift

[in] Slitamento dell'indicatore nel grafico prezzi.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iFrAMA.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iFrAMA."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
```

```
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- tracciamento iFrAMA
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // uso iFrAMA
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iFrAMA;             // tipo della funzione
input int                   ma_period=14;                 // periodo di media
input int                   ma_shift=0;                   // slittamento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // tipo del prezzo
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffer indicatore
double         iFrAMABuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iFrAMA
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Fractal Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- imposta lo slittamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
```

```
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[3];
      //--- periodo della media mobile
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- slittamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo di prezzo
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- tipo di prezzo
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iFrAMA per il simbol
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore iFrAMA
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iFrAMA
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iFrAMABuffer è maggiore del numero di valori dell'indicatore i
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array iFrAMABuffer con valori dell'indicatore Fractal Adaptive Moving
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Fractal Adaptive Moving Average
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
```

```
//| Riempie il buffer indicatore dall'indicatore iFrAMA                              |
//+----------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Fractal
                         int shift,         // slittamento
                         int ind_handle,    // handle dell'indicatore iFrAMA
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iFrAMABuffer con valori dal buffer indicatore che h
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iFrAMA, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+----------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                           |
//+----------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iGator

La funzione restituisce l'handle dell'indicatore Gator. L'oscillatore mostra la differenza tra le linee blu e rossa di Alligator (istogramma superiore) e la differenza tra le linee rossa e verde (istogramma inferiore).

```
int  iGator(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  jaw_period,        // periodo per il calcolo di jaws
   int                  jaw_shift,         // slittamento orizzontale jaws
   int                  teeth_period,      // periodo per il calcolo di teeth
   int                  teeth_shift,       // slittamento orizzontale teeth
   int                  lips_period,       // periodo per il calcolo di lips
   int                  lips_shift,        // slittamento orizzontale lips
   ENUM_MA_METHOD       ma_method,         // tipo di smussamento
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

jaw_period

[in] Periodo di mediazione per la linea blu (mascella dell'Alligatore, _* Alligator's Jaw).

jaw_shift

[in] Lo shift per la linea blu relativa al grafico dei prezzi. Non è direttamente collegato con lo slittamento visivo dell'istogramma dell'indicatore.

teeth_period

[in] Periodo medio per la linea rossa (Alligator's Teeth, _*denti dell'alligatore).

teeth_shift

[in] The shift of the red line relative to the price chart. Non è direttamente collegato con lo slittamento visivo dell'istogramma dell'indicatore.

lips_period

[in] Periodo medio per la linea verde (labbra dell'Alligatore, _*Alligator's lips).

lips_shift

[in] Lo slittamento della linea verde rispetto al grafico prezzi. Non è direttamente collegato con lo slittamento visivo dell'istogramma dell'indicatore.

ma_method

[in] Tipo di smussamento. Può essere uno dei valori di ENUM_MA_METHOD.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

Numeri di buffer: 0 -, UPPER_HISTOGRAM 1 - buffer colore dell'istogramma superiore, 2 - LOWER_HISTOGRAM, 3 - buffer colore dell'istogramma inferiore.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iGator.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iGator."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Gator Oscillator stand

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- tracciamento GatorUp
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- tracciamento GatorDown
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
```

```
  {
   Call_iGator,            // usa iGator
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iGator;       // tipo della funzione
input string                symbol=" ";             // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
input int                   jaw_period=13;          // periodo della linea Jaw
input int                   jaw_shift=8;            // slittamento della linea Jaw
input int                   teeth_period=8;         // periodo della linea Teeth
input int                   teeth_shift=5;          // slittamento della linea Teeth
input int                   lips_period=5;          // periodo della linea Lips
input int                   lips_shift=3;           // slittamento della linea Lips
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // medoto di media delle linee di A
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// tipo di prezzo utilizzato per
//--- buffers indicatore
double          GatorUpBuffer[];
double          GatorUpColors[];
double          GatorDownBuffer[];
double          GatorDownColors[];
//--- variabile per memorizzare l'handle dell'indicatore iGator
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- valori di slittamento per gli istogrammi superiore ed inferiore
int shift;
//--- manterremo il numero di valori nell'indicatore Gator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  Tutti gli slittamenti specificati nei parametri si riferiscono all'indicatore Alliga
  E' per questo che non spostano l'indicatore Gator in sé, ma spostano le linee Alliga
  i cui valori sono utilizzati per il calcolo del Gator Oscillator!
*/
//--- Calcoliamo lo slittamento per gli istogrammi superiore ed inferiore, che è pari
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
```

```
//--- nonostante l'indicatore contiene due istogrammi, viene utilzzato lo stesso lo s
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iGator)
      handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
                    lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[8];
      //--- pediodi e slittamenti delle linee dell' Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
      //--- tipo di smussamento
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- tipo di prezzo
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- crea l'handle
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iGator per il simbo
```

```
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Gator Oscillator
   short_name=StringFormat("iGator(%s/%s, %d, %d ,%d, %d, %d, %d)",name,EnumToString(p
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iGator
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array GatorUpBuffer è maggiore del numero di valori dell'indicatore
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
```

```
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore Gator Oscillator
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
      shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Gator Oscillator
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iGator                          |
//+------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],         // buffer indicatore per l'is
                           double &up_color_buffer[],    // buffer indicatore per gli
                           double &downs_buffer[],       // buffer indicatore per l'is
                           double &downs_color_buffer[], // buffer indicatore per i pi
                           int u_shift,                  // slittamento per l'istogram
                           int ind_handle,               // handle dell'indicatore iGa
                           int amount                    // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array GatorUpBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iGator, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array GatorUpColors con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iGator, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
```

```
//--- riempie una parte dell'array GatorDownBuffer con valori dal buffer indicatore ch
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iGator, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array GatorDownColors con valori dal buffer indicatore ch
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iGator, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iIchimoku

La funzione restituisce l'handle dell'indicatore Ichimoku Kinko Hyo.

```
int  iIchimoku(
   string           symbol,           // nome simbolo
   ENUM_TIMEFRAMES  period,           // periodo
   int              tenkan_sen,       // periodo di Tenkan-sen
   int              kijun_sen,        // periodo di Kijun-sen
   int              senkou_span_b     // periodo di Senkou Span B
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

tenkan_sen

[in] Periodo medio per Tenkan Sen.

kijun_sen

[in] Periodo medio per Kijun Sen.

senkou_span_b

[in] Periodo medio per Senkou Span B.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri di buffer: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_iIchimoku.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
```

```
#property description "dei buffers indicatore per l'indicatore tecnico iIchimoku."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri, proprio come nell' Ichimoku Kinko Hy

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- il tracciamento di Tenkan_sen
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- il tracciamento di Kijun_sen
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- il tracciamento di Senkou_Span
#property indicator_label3  "Senkou Span A;Senkou Span B" // due campi verranno mostra
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- il tracciamento di Chikou_Span
#property indicator_label4  "Chinkou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,         // uso iIchimoku
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation             type=Call_iIchimoku;       // tipo della funzione
input int                  tenkan_sen=9;              // periodo di Tenkan-sen
input int                  kijun_sen=26;              // periodo di Kijun-sen
input int                  senkou_span_b=52;          // periodo di Senkou Span B
input string               symbol=" ";                // simbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
```

```
double         Tenkan_sen_Buffer[];
double         Kijun_sen_Buffer[];
double         Senkou_Span_A_Buffer[];
double         Senkou_Span_B_Buffer[];
double         Chinkou_Span_Buffer[];
//--- variabile per memorizzare l'handle dell'indicatore iIchimoku
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Ichimoku Kinko Hyo
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
   SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
   SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- Imposta lo spostamento del canale Senkou Span delle barre kijun_sen in direzione
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- impostare uno slittamento per la linea Chikou Span non è necessario, in quanto i
//--- sono già memorizzati con uno slittamento in iIchimoku
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[3];
      //--- pediodi e slittamenti delle linee dell' Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=tenkan_sen;
```

```
      pars[1].type=TYPE_INT;
      pars[1].integer_value=kijun_sen;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=senkou_span_b;
      //--- crea l'handle
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iIchimoku per il sir
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Ichimoku Kinko Hyo
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iIchimoku
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'
```

```
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array Tenkan_sen_Buffer è maggiore del numero di valori dell'indicat
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore Ichimoku Kinko Hyo
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
      kijun_sen,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- manterremo il numero di valori dell'indicatore Ichimoku Kinko Hyo
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iIchimoku                             |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // buffer indicatore per l
                           double &kijun_sen_buffer[],      // buffer indicatore per l
                           double &senkou_span_A_buffer[],  // buffer indicatore della
                           double &senkou_span_B_buffer[],  // buffer indicatore della
                           double &chinkou_span_buffer[],   // buffer indicatore della
                           int senkou_span_shift,           // slittamento della linea
                           int ind_handle,                  // handle dell'indicatore
                           int amount                       // numero di valori copiat
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array Tenkan_sen_Buffer con valori dal buffer indicatore
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
```

```
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iIchimoku, codice eri
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array Kijun_sen_Buffer con valori dal buffer indicatore c
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iIchimoku, codice eri
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array Chinkou_Span_Buffer con valori dal buffer indicator
//--- se senkou_span_shift>0, la linea viene slittata in direzione futura di senkou_sp
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("3.Fallimento nel copiare i dati dall'indicatore iIchimoku, codice e
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array Senkou_Span_A_Buffer con valori dal buffer indicato
//--- se senkou_span_shift>0, la linea viene slittata in direzione futura di senkou_sp
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("4.Fallimento nel copiare i dati dall'indicatore iIchimoku, codice e
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array Senkou_Span_B_Buffer con valori dal buffer indicato
//--- quando copiamo Span Chinkou, non abbiamo bisogno di prendere in considerazione l
//--- sono già memorizzati con uno slittamento in iIchimoku
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("5.Fallimento nel copiare i dati dall'indicatore iIchimoku, codice e
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
```

```
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iBWMFI

La funzione restituisce l'handle dell'indicatore Market Facilitation Index. Ha un solo buffer.

```
int  iBWMFI(
   string               symbol,              // nome del simbolo
   ENUM_TIMEFRAMES      period,              // periodo
   ENUM_APPLIED_VOLUME  applied_volume       // tipo di volume per il calcolo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

applied_volume

[in] Il volume utilizzato. Può essere una delle costanti di ENUM_APPLIED_VOLUME.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_iBWMFI.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iBWMFI."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- il tracciamento di iBWMFI
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,            // uso iBWMFI
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iBWMFI;          // tipo di funzione
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;// tipo di volume
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double          iBWMFIBuffer[];
double          iBWMFIColors[];
//--- variabile per memorizzare l'handle dell'indicatore iBWMFI
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Market Facilitation Index by Bil
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
   else
```

```
      {
       //--- riempie la struttura con i parametri dell'indicatore
       MqlParam pars[1];
       //--- tipo di volume
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
      }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
      {
       //--- dice riguardo il fallimento e l'output del codice di errore
       PrintFormat("Fallimento nel creare l'handle dell'indicatore iBWMFI per il simbol
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- l'indicatore si ferma precocemente
       return(INIT_FAILED);
      }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Market Faci
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iBWMFI
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
       return(0);
      }
```

```
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iBWMFIBuffer è maggiore del numero di valori dell'indicatore
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array con i valori dell'indicatore Market Facilitation Index by Bill W
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- manterremo il numero di valori dell'indicatore Market Facilitation Index by Bill
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iBWMFI                                |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],    // buffer indicatore dei valori dell'
                           double &colors[],    // buffer indicatore dei colori dell'
                           int ind_handle,      // handle dell'indicatore iBWMFI
                           int amount           // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'indicatore iBWMFIBuffer con valori dal buffer indicatore
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iBWMFI, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
```

```
//--- riempie una parte dell'array iBWMFIColors con valori dal buffer indicatore che h
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iBWMFI, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iMomentum

La funzione restituisce l'handle del Momentum. Ha un solo buffer.

```
int  iMomentum(
   string                symbol,            // nome simbolo
   ENUM_TIMEFRAMES       period,            // periodo
   int                   mom_period,        // periodo medio
   ENUM_APPLIED_PRICE    applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

mom_period

[in] Periodo medio (conteggio barre) per il calcolo della variazione di prezzo.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                        Demo_iMomentum.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iMomentum."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Momentum standard."

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- tracciamento iMomentum
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,          // usa iMomentum
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iMomentum;       // tipo della funzione
input int                   mom_period=14;             // periodo di Momentum
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double          iMomentumBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iMomentum
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Momentum
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
```

```
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[2];
      //--- periodo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- tipo di prezzo
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iMomentum per il sir
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Momentum
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iMomentum
```

```
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iMomentumBuffer è maggiore del numero di valori dell'indicato
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array iMomentumBuffer con valori dell'indicatore Momentum
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero dei valori nell'indicatore Momentum
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iMomentum                             |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori Momentum
                         int ind_handle,    // handle dell'indicatore iMomentum
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iMomentumBuffer con valori dal buffer indicatore ch
```

```
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iMomentum, codice err
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iMFI

La funzione restituisce l'handle dell'indicatore Money Flow Index.

```
int  iMFI(
   string               symbol,             // nome del simbolo
   ENUM_TIMEFRAMES      period,             // periodo
   int                  ma_period,          // periodo medio
   ENUM_APPLIED_VOLUME  applied_volume      // tipo di volume per il calcolo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio (conteggio barre) per i calcoli.

applied_volume

[in] Il volume utilizzato. Può essere uno qualsiasi dei valori ENUM_APPLIED_VOLUME.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                             Demo_iMFI.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                             https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iMFI."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Money Flow Index stand

#property indicator_separate_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- il tracciamento di iMFI
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- livelli orizzontali nella finestra dell'indicatore
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // usa iMFI
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iMFI;              // tipo della funzione
input int                   ma_period=14;                // periodo
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;  // tipo di volume
input string                symbol=" ";                  // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // timeframe
//--- buffer indicatore
double         iMFIBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iBWMFI
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Money Flow Index
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
```

```
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[2];
      //--- periodo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo di volume
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iMFI per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Money Flow
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

```
  {
//--- numero di valori copiati dall' indicatore iMFI
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iMFIBuffer è maggiore del numero di valori dell'indicatore iMF
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array iMFIBuffer con i valori dell'indicatore Money Flow Index
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizziamo il numero di valori dell'indicatore Money Flow Indexx
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iMFI                            |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer indicatore dei valori di Money I
                         int ind_handle,    // handle dell'indicatore iMFI
                         int amount         // numero di valori copiati
                         )
  {
//--- resetta codice errore
```

```
   ResetLastError();
//--- riempie una parte dell'array iMFIBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iMFI, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iMA

La funzione restituisce l'handle dell'indicatore Moving Average. Ha un solo buffer.

```
int  iMA(
   string                symbol,            // nome simbolo
   ENUM_TIMEFRAMES       period,            // periodo
   int                   ma_period,         // periodo medio
   int                   ma_shift,          // slittamento orizzontale
   ENUM_MA_METHOD        ma_method,         // tipo di slittamento
   ENUM_APPLIED_PRICE    applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio per il calcolo della media mobile.

ma_shift

[in] Slittamento dell'indicatore rispetto al grafico dei prezzi.

ma_method

[in] Tipo di smussamento. Può essere uno dei valori ENUM_MA_METHOD.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                                  Demo_iMA.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iMA."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili a Moving Average."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iMA
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,               // usa iMA
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation             type=Call_iMA;               // tipo della funzione
input int                  ma_period=10;                // periodo di ma
input int                  ma_shift=0;                  // slittamento
input ENUM_MA_METHOD       ma_method=MODE_SMA;          // tipo di smussamento
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;   // tipo del prezzo
input string               symbol=" ";                  // simbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // timeframe
//--- buffer indicatore
double         iMABuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iMA
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
```

```
//--- imposta lo slittamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[4];
      //--- periodo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- slittamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo di smussamento
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo di prezzo
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iMA per il simbolo %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Moving Average
   short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period, ma_shift,EnumToString(ma_method),EnumToString(ap
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

```
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iMA
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iMABuffer è maggiore del numero di valori dell'indicatore iMA
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array iMABuffer con valori dell'indicatore Adaptive Moving Average
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

```
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Moving Average
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore MA                              |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // buffer indicatore dei valori di Moving
                         int shift,          // slittamento
                         int ind_handle,     // handle dell'indicatore iMA
                         int amount          // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iMABuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iMA, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come n
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iOsMA

La funzione restituisce l'handle dell'indicatore Moving Average of Oscillator. L'oscillatore OsMA mostra la differenza tra i valori di MACD e la sua linea di segnale. Ha un solo buffer.

```
int  iOsMA(
   string              symbol,              // nome simbolo
   ENUM_TIMEFRAMES     period,              // periodo
   int                 fast_ema_period,     // periodo per Fast Moving Average
   int                 slow_ema_period,     // periodo per Slow Moving Average
   int                 signal_period,       // periodo per la loro differenza
   ENUM_APPLIED_PRICE  applied_price        // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

fast_ema_period

[in] Periodo per il calcolo di Fast Moving Average.

slow_ema_period

[in] Periodo per il calcolo di Slow Moving Average.

signal_period

[in] Periodo per il calcolo della linea Signal.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

In alcuni sistemi questo oscillatore è anche conosciuto come istogramma MACD.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iOsMA.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                     https://www.mql5.com |
```

```
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iOsMA."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Moving Average of Osci

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iOsMA
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,             // uso iOsMA
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iOsMA;           // tipo della funzione
input int                   fast_ema_period=12;        // periodo di fast ma
input int                   slow_ema_period=26;        // periodo di slow ma
input int                   signal_period=9;           // periodo medio della differenz
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double         iOsMABuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iAMA
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
```

```
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[4];
      //--- periodo di fast ma
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- periodo di slow ma
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- periodo di media di differenza tra media mobile veloce e lenta
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- tipo di prezzo
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iOsMA per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Moving Average of (
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
```

```
                              fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iOsMA
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iOsMABuffer è maggiore del numero di valori dell'indicatore iO
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore iOsMA
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

```
                              short_name,
                              values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Moving Average of Oscillator
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iOsMA                                 |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // buffer indicatore dei valori OsMA
                         int ind_handle,        // handle dell'indicator iOsMA
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iOsMABuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iOsMA, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iMACD

La funzione restituisce l'handle dell' indicatore Moving Averages Convergence/Divergence. Nei sistemi in cui OsMA viene chiamato istogramma MACD, questo indicatore viene visualizzato come due linee. Nel terminale client le Medie Mobili di Convergenza/Divergenza si mostrano come un istogramma.

```
int  iMACD(
   string               symbol,              // nome simbolo
   ENUM_TIMEFRAMES      period,              // periodo
   int                  fast_ema_period,     // periodo per il calcolo di media Fast
   int                  slow_ema_period,     // periodo per il calcolo di media Slow
   int                  signal_period,       // periodo per la loro differenza media
   ENUM_APPLIED_PRICE   applied_price        // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

fast_ema_period

[in] Periodo per il calcolo di Fast Moving Average.

slow_ema_period

[in] Periodo per il calcolo di Slow Moving Average.

signal_period

[in] Periodo per il calcolo della linea Signal.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri del buffer sono i seguenti: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                              Demo_iMACD.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
```

```
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iMACD."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili a MACD standard."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- il tracciamento di MACD
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- il tracciamento di Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,              // uso iMACD
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iMACD;           // tipo della funzione
input int                   fast_ema_period=12;        // periodo di fast ma
input int                   slow_ema_period=26;        // periodo di slow ma
input int                   signal_period=9;           // periodo medio della differenz
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffers indicatore
double         MACDBuffer[];
double         SignalBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iMACD
int    handle;
//--- variabile per memorizzare
```

```
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Moving Averages Convergence/Dive
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[4];
      //--- periodo di fast ma
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- periodo di slow ma
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- periodo di media di differenza tra media mobile veloce e lenta
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- tipo di prezzo
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
```

```
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iMACD per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore Moving Average Conv
   short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iMACD
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array MACDBuffer è maggiore del numero di valori dell'indicatore iMA
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
```

```
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore iMACD
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Moving Averages indicator Converge
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iMACD                           |
//+------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // buffer indicatore di valori MA
                           double &signal_buffer[],  // buffer indicatore della linea
                           int ind_handle,           // handle dell'indicatore iMACD
                           int amount                // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iMACDBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iMACD, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }

//--- riempie una parte dell'array SignalBuffer con valori dal buffer indicatore che h
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iMACD, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iOBV

La funzione restituisce l'handle dell' indicatore On Balance Volume. Ha un solo buffer.

```
int  iOBV(
   string                symbol,             // nome simbolo
   ENUM_TIMEFRAMES       period,             // periodo
   ENUM_APPLIED_VOLUME   applied_volume      // tipo di volume per il calcolo
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

applied_volume

[in] Il volume utilizzato. Può essere uno qualsiasi dei valori ENUM_APPLIED_VOLUME.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iOBV.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iOBV."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- tracciamento di iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,             // uso iOBV
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iOBV;               // tipo della funzione
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo di volume
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffers indicatore
double          iOBVBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iOBV
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore On Balance Volume
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
```

```
      MqlParam pars[1];
      //--- tipo di volume
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iOBV per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- Mostra il simbolo/timeframe per cui è calcolato l'indicatore On Balance Volume
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                  |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iOBV
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
```

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iOBVBuffer è maggiore del numero di valori dell'indicatore iOB
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore OBV
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizziamo il numero di valori dell'indicatore On Balance Volume
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iOBV                                  |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // buffer indicatore dei valori OBV
                         int ind_handle,        // handle dell'indicatore iOBV
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iOBVBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iOBV, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
```

```
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iSAR

La funzione restituisce l'handle dell'indicatore Parabolic Stop and Reverse system. Ha un solo buffer.

```
int  iSAR(
   string           symbol,      // nome simbolo
   ENUM_TIMEFRAMES  period,      // periodo
   double           step,        // step di incremento
   double           maximum      // livello massimo di stop
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

step

[in] L'incremento di livello di step, solitamente 0.02.

maximum

[in] Il livello di stop massimo, di solito 0.2.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iSAR.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iSAR."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Parabolic Stop and Rev

#property indicator_chart_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- tracciamento di iSAR
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // usa iSAR
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iSAR;              // tipo della funzione
input double                step=0.02;                   // step - il fattore di accel
input double                maximum=0.2;                 // valore massimo degli step
input string                symbol=" ";                  // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // timeframe
//--- buffers indicatore
double          iSARBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iSAR
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Parabolic SAR
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- imposta un codice simbolo dal set di caratteri Wingdings per la proprietà PLOT_A
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
```

```
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[2];
      //--- valore step
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- limite del valore di step che può essere utilizzato per i calcoli
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iSAR per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Parabolic SAR
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

```
//--- numero di valori copiati dall' indicatore iSAR
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iSARBuffer è maggiore del numero di valori dell'indicatore iSA
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore iSAR
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Parabolic SAR
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iSAR                            |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // buffer indicatore dei valori di Par
                         int ind_handle,        // handle dell'indicatore iSAR
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
```

```
//--- riempie una parte dell'array iSARBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iSAR, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iRSI

La funzione restituisce l'handle dell'indicatore Relative Strength Index. Ha un solo buffer.

```
int  iRSI(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Il periodo medio per il calcolo dell'indicatore.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iRSI.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iRSI."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Relative Strength Inde

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- tracciamento di iRSI
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- limiti per la visualizzazione dei valori nella finestra dell'indicatore
#property indicator_maximum 100
#property indicator_minimum 0
//--- livelli orizzontali nella finestra dell'indicatore
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // usa iRSI
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iRSI;               // tipo della funzione
input int                   ma_period=14;                 // periodo di media
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // tipo del prezzo
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffer indicatore
double          iRSIBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iRSI
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Relative Strength Index
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
```

```
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[2];
      //--- periodo della media mobile
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- limite del valore di step che può essere utilizzato per i calcoli
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iRSI per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Relative St
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iRSI
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iRSIBuffer è maggiore del numero di valori dell'indicatore iRS
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore RSI
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizziamo il numero di valori dell'indicatore Relative Strength Index
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iRSI                            |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rsi_buffer[],  // buffer indicatore dei valori di Rel
                         int ind_handle,        // handle dell'indicatore iRSI
```

```
                           int amount              // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iRSIBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iRSI, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iRVI

La funzione restituisce l'handle dell'indicatore Relative Vigor Index.

```
int  iRVI(
   string           symbol,        // nome simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Il periodo medio per il calcolo dell'indicatore RVI.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri del buffer sono i seguenti: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                              Demo_iRVI.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iRVI."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Relative Vigor Index s

#property indicator_separate_window
#property indicator_buffers 2
```

```
#property indicator_plots   2
//--- il tracciamento di RVI
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- il tracciamento di Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // uso iRVI
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation             type=Call_iRVI;              // tipo della funzione
input int                  ma_period=10;                // periodo per i calcoli
input string               symbol=" ";                  // simbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // timeframe
//--- buffers indicatore
double         RVIBuffer[];
double         SignalBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iRVI
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Relative Vigor Index
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
```

```
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[1];
      //--- periodo per i calcoli
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iRVI per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Relative V
   short_name=StringFormat("iRVI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

```
  {
//--- numero di valori copiati dall' indicatore RVI
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array RVIBuffer è maggiore del numero di valori dell'indicatore iRVI
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore iRVI
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è anc
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizziamo il numero di valori dell'indicatore Relative Vigor Index
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iRVI                            |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // buffer indicatore dei valori Rel
                         double &signal_buffer[],  // buffer indicatore della linea di
                         int ind_handle,           // handle dell'indicatore iRVI
                         int amount                // numero dei valori copiati
                         )
  {
```

```
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iRVIBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iRVI, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- riempie una parte dell'array SignalBuffer con valori dal buffer indicatore che h
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iRVI, codice errore %
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iStdDev

La funzione restituisce l'handle dell'indicatore Standard Deviation. Ha un solo buffer.

```
int  iStdDev(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   int                  ma_shift,          // slittamento orizzontale
   ENUM_MA_METHOD       ma_method,         // tipo di smussamento
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio per il calcolo degli indicatori.

ma_shift

[in] Slittamento dell'indicatore rispetto al grafico dei prezzi.

ma_method

[in] Tipo della media. Può essere uno qualsiasi dei valori ENUM_MA_METHOD.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_iStdDev.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iStdDev."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili allo Standard Deviation s

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iStdDev
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,           // usa iStdDev
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iStdDev;         // tipo della funzione
input int                   ma_period=20;              // periodo di media
input int                   ma_shift=0;                // slittamento
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // tipo di smussamento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double         iStdDevBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iStdDev
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Standard Deviation
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
```

```
//--- imposta lo slittamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[4];
      //--- periodo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- slittamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo di smussamento
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo di prezzo
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iStdDev per il simbo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Standard Deviation
   short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(ma_method),EnumToString(app
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

```
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iStdDev
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iStdDevBuffer è maggiore del numero di valori dell'indicatore
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore Standard Deviation
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

```
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Standard Deviation
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iStdDev                         |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // buffer indicatore della linea di St
                         int std_shift,         // slittamento della linea di Standard
                         int ind_handle,        // handle dell'indicatore iStdDev
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iStdDevBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iStdDev, codice erro
      //--- esce con risultato zero - significa che l'indicatore è considerato come n
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iStochastic

La funzione restituisce l'handle dell'indicatore Stochastic Oscillator.

```
int  iStochastic(
   string           symbol,          // nome simbolo
   ENUM_TIMEFRAMES  period,          // periodo
   int              Kperiod,         // K-period (numero di barre per il calcolo)
   int              Dperiod,         // D-period (periodo del primo slittamento)
   int              slowing,         // slittamento finale
   ENUM_MA_METHOD   ma_method,       // tipo di slittamento
   ENUM_STO_PRICE   price_field      // metodo di calcolo stocastico
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

Kperiod

[in] Periodo medio (conteggio barre) per il calcolo linea %K.

Dperiod

[in] Periodo medio (conteggio barre) per il calcolo linea %D.

slowing

[in] Valore Slowing.

ma_method

[in] Tipo della media. Può essere uno qualsiasi dei valori ENUM_MA_METHOD.

price_field

[in] Parametro di selezione prezzo per i calcoli. Può essere uno dei valori ENUM_STO_PRICE.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Nota

I numeri di buffer: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                      Demo_iStochastic.mq5 |
```

```
//|                                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                                 https://www.mql5.com |
//+------------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iStochastic."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili allo Stochastic Oscillato

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- il tracciamento Stochastic
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- il tracciamento di Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- imposta il limite dei valori degli indicatori
#property indicator_minimum 0
#property indicator_maximum 100
//--- livelli orizzontali nella finestra dell'indicatore
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // usa iStochastic
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation             type=Call_iStochastic;     // tipo della funzione
input int                  Kperiod=5;                 // il periodo K (il numero di ba
input int                  Dperiod=3;                 // il periodo D (il periodo di s
input int                  slowing=3;                 // periodo di smussamento finale
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // tipo di smussamento
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;   // metodo per il calcolo di Sto
```

```
input string               symbol=" ";                // simbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffers indicatore
double         StochasticBuffer[];
double         SignalBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iStochastic
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Stochastic Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array di buffer indicatori
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[5];
      //--- il periodo K per i calcoli
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- il periodo D per lo smussamento primario
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- il periodo K per lo smussamento finale
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
      //--- tipo di smussamento
```

```
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ma_method;
      //--- metodo di calcolo dello Stochastic
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iStochastic per il s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Stochastic Oscillat
   short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString
                           Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iStochastic
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'
```

```
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array StochasticBuffer è maggiore del numero di valori dell'indicato
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore iStochastic
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizzare il numero di valori dell'indicatore Stochastic Oscillator
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iStochastic                           |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // buffer indicatore dei valori d
                           double &signal_buffer[],  // buffer indicatore della linea
                           int ind_handle,           // handle dell'indicatore iStocha
                           int amount                // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iBullsPowerBuffer con valori dal buffer indicatore
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iStochastic, codice e
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- riempie una parte dell'array SignalBuffer con valori dal buffer indicatore che h
```

```
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iStochastic, codice e
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iTEMA

La funzione restituisce l'handle dell'indicatore Triple Exponential Moving Average. Ha un solo buffer.

```
int  iTEMA(
   string               symbol,           // nome del simbolo
   ENUM_TIMEFRAMES      period,           // periodo
   int                  ma_period,        // periodo medio
   int                  ma_shift,         // slittamento orizzontale dell'indicatore
   ENUM_APPLIED_PRICE   applied_price     // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio (conteggio barre) per il calcolo.

ma_shift

[in] Slittamento dell'indicatore relativo al grafico dei prezzi.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                              Demo_iTEMA.mq5 |
//|                                 Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iTEMA."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
```

```
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili al Triple Exponential Mov

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iTEMA
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,              // uso iTEMA
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iTEMA;           // tipo della funzione
input int                   ma_period=14;              // periodo di media
input int                   ma_shift=0;                // slittamento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double         iTEMABuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iTEMA
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Triple Exponential Moving Averag
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato  |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- imposta lo slittamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
```

```
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[3];
      //--- periodo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- slittamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo di prezzo
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iTEMA per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Triple Exponential
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iTEMA
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iTEMABuffer è maggiore del numero di valori dell'indicatore iT
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array con valori dell'indicatore Triple Exponential Moving Average
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Triple Exponential Moving Average
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
```

```
//| Riempie il buffer indicatore dall'indicatore iTEMA |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // buffer indicatore dei valori di Tri
                         int t_shift,           // slittamento della linea
                         int ind_handle,        // handle dell'indicatore iTEMA
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iBullsPowerBuffer con valori dal buffer indicatore
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iTEMA, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iTriX

La funzione restituisce l'handle dell'indicatore Triple Exponential Moving Averages Oscillator. Ha un solo buffer.

```
int  iTriX(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  ma_period,         // periodo medio
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

ma_period

[in] Periodo medio (conteggio barre) per i calcoli.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                              Demo_iTriX.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                        https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iTriX."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iTriX
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,             // uso iTriX
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation             type=Call_iTriX;           // tipo della funzione
input int                  ma_period=14;              // periodo
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo di prezzo
input string               symbol=" ";                // simbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double         iTriXBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iTriX
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Triple Exponential Moving Averag
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                             |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
```

```
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[2];
      //--- periodo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo di prezzo
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iTriX per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Triple Exponential
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iTriX
```

```
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iTriXBuffer è maggiore del numero di valori dell'indicatore iT
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array con valori dell'indicatore Triple Exponential Moving Averages (
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell'indicatore Triple Exponential Moving Averages
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iTriX                                 |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // buffer indicatore dei valori di Tri
                         int ind_handle,        // handle dell'indicatore iTriX
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iBullsPowerBuffer con valori dal buffer indicatore
```

```
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iTriX, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                       |
//+------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iWPR

La funzione restituisce l'handle di indicatore Larry Williams' Percent Range. Ha un solo buffer.

```
int  iWPR(
   string           symbol,          // nome simbolo
   ENUM_TIMEFRAMES  period,          // periodo
   int              calc_period      // periodo medio
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

calc_period

[in] Periodo (conteggio barre) per i calcoli dell'indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                            Demo_iWPR.mq5 |
//|                                  Copyright 2011, MetaQuotes Software Corp. |
//|                                                       https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iWPR."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iWPR
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- imposta il limite dei valori degli indicatori
#property indicator_minimum -100
#property indicator_maximum 0
//--- livelli orizzontali nella finestra dell'indicatore
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // usa iWPR
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iWPR;           // tipo della funzione
input int                   calc_period=14;           // periodo
input string                symbol=" ";               // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;    // timeframe
//--- buffer indicatore
double          iWPRBuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iWPR
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Larry Williams' Percent Range
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
```

```
//--- crea l'handle dell'indicatore
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[1];
      //--- periodo
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iWPR per il simbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore tecnico Williams' I
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                              |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iWPR
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
```

```
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iWPRBuffer è maggiore del numero di valori dell'indicatore iWP
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l'array con i valori dell'indicatore Williams' Percent Range
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- manterremo il numero di valori dell'indicatore Larry Williams' Percent Range
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iWPR                            |
//+------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[],  // buffer indicatore dei valori di Wil
                         int ind_handle,        // handle dell'indicatore iWPR
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iWPRBuffer con valori dal buffer indicatore che ha
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iBearsPower, codice e
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
```

```
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iVIDyA

La funzione restituisce l'handle dell'indicatore Variable Index Dynamic Average. Ha un solo buffer.

```
int  iVIDyA(
   string               symbol,            // nome del simbolo
   ENUM_TIMEFRAMES      period,            // periodo
   int                  cmo_period,        // periodo per il Chande Momentum
   int                  ema_period,        // periodo di smussamento EMA
   int                  ma_shift,          // slittamento orizzontale sul grafico dei p
   ENUM_APPLIED_PRICE   applied_price      // tipo di prezzo o handle
   );
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

cmo_period

[in] Periodo (conteggio barre) per il calcolo di Chande Momentum Oscillator.

ema_period

[in] Periodo EMA (conteggio barre) per i calcoli del fattore si smussamento.

ma_shift

[in] Slittamento dell'indicatore rispetto al grafico dei prezzi.

applied_price

[in] Il prezzo utilizzato. Può essere una qualsiasi delle costanti ENUM_APPLIED_PRICE o un handle di un altro indicatore.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                          Demo_iVIDyA.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                                      https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iVIDyA."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i
#property description "Tutti gli altri parametri sono simili a Variable Index Dynamic

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- il tracciamento di iVIDyA
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                          |
//+------------------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,             // uso iVIDyA
   Call_IndicatorCreate     // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iVIDyA;          // tipo della funzione
input int                   cmo_period=15;             // il periodo Chande Momentum
input int                   ema_period=12;             // periodo per il fattore di smu
input int                   ma_shift=0;                // slittamento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo di prezzo
input string                symbol=" ";                // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer indicatore
double         iVIDyABuffer[];
//--- variabile per memorizzare l'handle dell'indicatore iVIDyA
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori nell' indicatore Variable Index Dynamic Average
int    bars_calculated=0;
//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato                       |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
```

```
//--- imposta lo slittamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- riempie la struttura con i parametri dell'indicatore
      MqlParam pars[4];
      //--- il periodo Chande Momentum
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- periodo per il fattore di smussamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- slittamento
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- tipo di prezzo
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iVIDyA per il simbol
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Variable Index Dyna
   short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
                           cmo_period,ema_period,ma_shift,EnumToString(applied_price))
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

```
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato                                    |
//+------------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iVIDyA
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'i
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iVIDyABuffer è maggiore del numero di valori dell'indicatore i
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie l' array con valori dell'indicatore Variable Index Dynamic Average
//--- se FillArrayFromBuffer restituisce false, significa che l'informazione non è and
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

```
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero di valori nell' indicatore Variable Index Dynamic Average
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iVIDyA                                |
//+------------------------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// buffer indicatore dei valodi ri Va
                         int v_shift,           // slittamento della linea
                         int ind_handle,        // handle dell'indicatore iVIDyA
                         int amount             // numero di valori copiati
                         )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iVIDyABuffer con valori dal buffer indicatore che h
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iVIDyA, codice errore
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore                                             |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

iVolumes

La funzione restituisce l'handle dell'indicatore Volumes. Ha un solo buffer.

```
int  iVolumes(
   string               symbol,             // nome del simbolo
   ENUM_TIMEFRAMES      period,             // periodo
   ENUM_APPLIED_VOLUME  applied_volume      // tipo di volume per il calcolo
   )
```

Parametri

symbol

[in] Il nome del simbolo dello strumento finanziario, i cui dati devono essere utilizzati per calcolare l'indicatore. Il valore NULL significa il simbolo corrente.

period

[in] Il valore del periodo può essere uno dei valori ENUM_TIMEFRAMES, 0 significa il corrente timefreame.

applied_volume

[in] Il volume usato. Può essere uno qualsiasi dei valori ENUM_APPLIED_VOLUME.

Valore restituito

Restituisce l'handle di un indicatore tecnico specificato, in caso di fallimento restituisce INVALID_HANDLE. La memoria del computer può essere liberata da un indicatore che non è più utilizzato, utilizzando la funzione IndicatorRelease(), al quale l'handle indicatore viene passato.

Esempio:

```
//+------------------------------------------------------------------------------+
//|                                                    Demo_iVolumes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. | //| |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicatore dimostra come ottenere i dati"
#property description "dei buffers indicatore per l'indicatore tecnico iVolumes."
#property description "Il simbolo e timeframe usati per i calcoli dell'indicatore,"
#property description "vengono impostati dai parametri simbolo e periodo."
#property description "Il metodo per la creazione dell'handle è impostato attraverso i

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- il tracciamento iVolumes
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------------------------+
//| Enumerazione dei metodi della creazione dell'handle                                |
//+------------------------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,          // uso iVolumes
   Call_IndicatorCreate    // usa IndicatorCreate
  };
//--- parametri di input
input Creation              type=Call_iVolumes;           // tipo della funzione
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo di volume
input string                symbol=" ";                   // simbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffers indicatore
double          iVolumesBuffer[];
double          iVolumesColors[];
//--- variabile per memorizzare l'handle dell'indicatore iVolumes
int    handle;
//--- variabile per memorizzare
string name=symbol;
//--- nome dell'indicatore sul grafico
string short_name;
//--- manterremo il numero di valori dell'indicatore Volumes
int    bars_calculated=0;
//+------------------------------------------------------------------------------------+
//| Funzione di inizializzazione Indicatore Personalizzato  |
//+------------------------------------------------------------------------------------+
int OnInit()
  {
//--- assegnazione di array al buffer indicatore
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- determinare il simbolo per cui viene disegnato l'indicatore
   name=symbol;
//--- eliminare gli spazi a destra e a sinistra
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se il risultato è zero nella lunghezza della stringa 'name'
   if(StringLen(name)==0)
     {
      //--- prende il simbolo del grafico indicatore a cui è attaccato
      name=_Symbol;
     }
//--- crea l'handle dell'indicatore
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
   else
```

```
      {
       //--- riempie la struttura con i parametri dell'indicatore
       MqlParam pars[1];
       //--- tipo di prezzo
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
      }
//--- se l'handle non viene creato
   if(handle==INVALID_HANDLE)
     {
      //--- dice riguardo il fallimento e l'output del codice di errore
      PrintFormat("Fallimento nel creare l'handle dell'indicatore iVolumes per il simb
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- l'indicatore si ferma precocemente
      return(INIT_FAILED);
     }
//--- mostra il simbolo/timeframe per cui è calcolato l'indicatore Volumes
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inizializzazione normale dell'indicatore
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione di iterazione indicatore personalizato |
//+------------------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- numero di valori copiati dall' indicatore iVolumes
   int values_to_copy;
//--- determina il numero di valori calcolati in dell'indicatore
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() returned %d, error code %d",calculated,GetLastErro
      return(0);
     }
//--- se è il primo inizio del calcolo dell'indicatore o se il numero di valori dell'
```

```
//--- o se è necessario calcolare l'indicatore per due o più barre (significa che qual
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se l' array iVolumesBuffer è maggiore del numero di valori dell'indicatore
      //--- altrimenti, copiamo meno della grandezza del buffer indicatore
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- vuol dire che non è la prima volta del calcolo dell'indicatore, e dopo l'u
      //--- per il calcolo non viene aggiunta più di una barra
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- riempie gli array con valori dell'indicatore iVolumes
//--- se FillArraysFromBuffer restituisce false, significa che l'informazione non è ar
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- forma il messaggio
   string comm=StringFormat("%s ==>  Valore aggiornato nell'indicatore %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- visualizza il messaggio di servizio sul grafico
   Comment(comm);
//--- memorizza il numero dei valori nell'indicatore Volumes
   bars_calculated=calculated;
//--- restituisce il valore prev_calculated per la chiamata successiva
   return(rates_total);
  }
//+------------------------------------------------------------------------------------+
//| Riempie il buffer indicatore dall'indicatore iVolumes |
//+------------------------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // buffer indicatore dei valori
                           double &color_buffer[],    // buffer indicatore dei colori
                           int ind_handle,            // handle dell'indicatore iVolum
                           int amount                 // numero di valori copiati
                           )
  {
//--- resetta codice errore
   ResetLastError();
//--- riempie una parte dell'array iVolumesBuffer con valori dal buffer indicatore che
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iVolumes, codice erro
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- riempie una parte dell'array iVolumesColors con valori dal buffer indicatore che
```

```
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- se la copia fallisce, dice il codice dell'errore
      PrintFormat("Fallimento nel copiare i dati dall'indicatore iVolumes, codice erro
      //--- esce con risultato zero - significa che l'indicatore è considerato come no
      return(false);
     }
//--- tutto è ok
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione indicatore |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- cancella il grafico dopo aver eliminato l'indicatore
   Comment("");
  }
```

Lavorare con i risultati di ottimizzazione

Funzioni per organizzare processi personalizzati dei risultati di ottimizzazione nello strategy tester. Esse possono essere chiamate durante l'ottimizzazione degli agenti di testing, così come a livello locale negli Expert Advisor e script.

Quando si esegue un Expert Advisor nello strategy tester, è possibile creare in proprio array di dati in base ai tipi semplici o strutture semplici (esse non contengono stringhe, oggetti di classi o di oggetti di array dinamici). Questo set di dati può essere salvato con la funzione FrameAdd() di una struttura speciale chiamata frame. Durante l'ottimizzazione di un Expert Advisor, ogni agente può inviare una serie di frames al terminale. Tutti i frame ricevuti vengono scritti nel file *.MQD nella cartella expert, terminal_directory/MQL5/Files/Tester denominata come l'Expert Advisor. Sono scritti nell'ordine in cui sono ricevuti dagli agenti. Ricezione di un frame nel terminale client da un agente di test genera l'evento TesterPass.

I frames possono essere memorizzati nella memoria del computer e in un file con il nome specificato. Il linguaggio MQL5 non pone limiti al numero di frames.

Limiti di memoria e spazio su disco nella MQL5 Cloud Network

La seguente limitazione si applica alle ottimizzazioni eseguite nella MQL5 Cloud Network: l'Expert Advisor non deve scrivere su disco più di 4GB di informazioni o utilizzare più di 4GB di RAM. Se il limite viene superato, l'agente di rete non sarà in grado di completare correttamente il calcolo e non riceverai il risultato. Tuttavia, ti verrà addebitato tutto il tempo speso per i calcoli.

Se hai bisogno di ottenere informazioni da ogni passaggio di ottimizzazione, invia dei frames senza scrivere sul disco. Per evitare di utilizzare operazioni con i file negli Expert Advisors durante i calcoli nella rete cloud MQL5, è possibile utilizzare il seguente controllo:

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Funzione	Azione
FrameFirst	Sposta un puntatore di lettura frame all'inizio e resetta il filtro impostato precedentemente
FrameFilter	Imposta il frame di lettura filtro e sposta il puntatore all'inizio
FrameNext	Legge un frame e sposta il puntatore al successivo

Funzione	Azione
FrameInputs	Riceve i parametri di input, in cui il frame viene formato
FrameAdd	Aggiunge un frame con dati
ParameterGetRange	Riceve i dati sulla gamma di valori ed lo step del cambiamento per una variabile di input quando si ottimizza un Expert Advisor nello Strategy Tester
ParameterSetRange	Specifica l'uso della variabile di input quando si ottimizza un Expert Advisor nello Strategy Tester: valore, step del cambiamento, i valori iniziali e finali

Vedi anche

Statistiche di Testing, Proprietà di un Programma MQL5 in esecuzione

FrameFirst

Consente di spostare il puntatore del frame leggendo dall'inizio e resetta un filtro settato .

```
bool  FrameFirst();
```

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

FrameFilter

Imposta il filtro di lettura del frame e sposta il puntatore all'inizio.

```
bool  FrameFilter(
   const string  name,          // Nome pubblico/etichetta
   long          id             // ID pubblico
   );
```

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Se una stringa vuota viene passata come primo parametro, il filtro funziona solo con un parametro numerico, cioè solo i frames con l'ID specificato verranno visualizzati. Se il valore del secondo parametro è ULONG_MAX, solo un filtro di testo funziona.

La chiamata di FrameFilter("", ULONG_MAX) è equivalente alla chiamata FrameFirst(), cioè pari a non utilizzare alcun filtro.

FrameNext

Legge un frame e sposta il puntatore al successivo. Ci sono due varianti della funzione.

1. Chiamata a ricevere un valore numerico

```
bool  FrameNext(
   ulong&   pass,      // Il numero di un pass nell'ottimizzazione, durante il quale è
   string&  name,      // Nome pubblico/etichetta
   long&    id,        // ID pubblico
   double&  value      // Valore
   );
```

2. Chiamata per ricevere tutti i dati di un frame

```
bool  FrameNext(
   ulong&   pass,      // Il numero di un pass nell'ottimizzazione, durante il quale è
   string&  name,      // Nome pubblico/etichetta
   long&    id,        // ID pubblico
   double&  value,     // Valore
   void&    data[]     // Array di un qualsiasi tipo
   );
```

Parametri

pass

[out] Il numero di un passaggio durante l'ottimizzazione nel tester strategia.

name

[out] Il nome dell'identificatore.

id

[out] Il valore dell'identificatore.

valore

[out] Un singolo valore numerico.

data

[out] Un array di un qualunque tipo.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Nella seconda versione della chiamata, è necessario gestire correttamente i dati ricevuti nell' array *data[]*.

FrameInputs

Riceve parametri di input, in cui è formato il frame con il numero di passo specificato.

```
bool  FrameInputs(
   ulong    pass,                // Il numero di un passo nell'ottimizzazione
   string&  parameters[],        // Un array di stringhe di tipo "parametroN=valoreN"
   uint&    parameters_count     // Il numero totale dei parametri
   );
```

Parametri

pass

[in] Il numero di un passaggio durante l'ottimizzazione nel tester strategia.

parameters

[out] Un array di stringhe con la descrizione dei nomi e dei valori dei parametri

parameters_count

[out] Il numero di elementi dell'array *parameters[]*.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

Nota

Dopo aver ottenuto il numero di stringhe *parameters_count* nell' array *parameters[],*, è possibile organizzare un ciclo per andare attraverso tutti i record. Questo ti aiuterà a trovare i valori dei parametri di ingresso di un Expert Advisor per il numero di passo specificato.

FrameAdd

Consente di aggiungere un frame con i dati. Ci sono due varianti della funzione.

1. Aggiunta di dati da un file

```
bool  FrameAdd(
   const string  name,         // Nome pubblico/etichetta
   long          id,           // ID pubblico
   double        value,        // Valore
   const string  filename      // Nome del file di dati
   );
```

2. Aggiunta di dati da un array di qualsiasi tipo

```
bool  FrameAdd(
   const string  name,         // Nome pubblico/etichetta
   long          id,           // ID pubblico
   double        value,        // Valore
   const void&   data[]        // Array di un qualsiasi tipo
   );
```

Parametri

name

[in] Etichetta frame pubblico. Essa può essere utilizzata per un filtro nella funzione FrameFilter().

id

[in] Un identificatore pubblico del frame. Essa può essere utilizzata per un filtro nella funzione FrameFilter().

valore

[in] Un valore numerico da scrivere nel frame. Esso è utilizzato per trasmettere un singolo risultato come nella funzione OnTester().

filename

[in] Il nome del file che contiene i dati da aggiungere al frame. Il file deve essere localizzao nella cartella MQL5/Files.

data

[in] Un array di qualsiasi tipo da scrivere nel frame. Passato per riferimento.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ottenere informazioni sull'errore, chiamare la funzione GetLastError().

ParameterGetRange

Riceve i dati sulla gamma di valori e lo step di cambiamento per una variabile di input quando si ottimizza un Expert Advisor nello Strategy Tester. Ci sono due varianti della funzione.

1. Ricezione di dati per il parametro di input di tipo integer

```
bool  ParameterGetRange(
   const string  name,           // nome parametro (input variable)
   bool&         enable,         // ottimizzazione parametro abilitata
   long&         value,          // valore parametro
   long&         start,          // valore iniziale
   long&         step,           // step di cambiamento
   long&         stop            // valore finale
   );
```

2. Ricezione di dati per il parametro di input di tipo reale

```
bool  ParameterGetRange(
   const string  name,           // nome parametro (input variable)
   bool&       enable,           // ottimizzazione parametro abilitata
   double&       value,          // valore parametro
   double&       start,          // valore iniziale
   double&       step,           // step di cambiamento
   double&       stop            // valore finale
   );
```

Parametri

name

[in] input variable ID. Queste variabili sono parametri esterni di un'applicazione. I loro valori possono essere specificati quando si avviano su un grafico o in un singolo test.

enable

[out] Flag di questo parametro che può essere utilizzato per enumerare i valori durante l'ottimizzazione dello Strategy Tester.

valore

[out] Valore parametro.

start

[out] Valore iniziale del parametro durante l'ottimizzazione.

step

[out] Step di modifica dei parametri durante l'enumerazione dei suoi valori.

stop

[out] Valore del parametro finale durante l'ottimizzazione.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

La funzione può essere chiamata solo dagli handlers OnTesterInit(), OnTesterPass() ed OnTesterDeinit(). E' stato introdotto per ricevere valori di parametri di input degli Expert Advisor e gli intervalli di variazione durante l'ottimizzazione nello Strategy Tester.

Quando viene chiamato in OnTesterInit(), i dati ottenuti possono essere utilizzati per ridefinire le regole per l'enumerazione di qualsiasi variabile di input utilizzando la funzione ParameterSetRange (). Pertanto, nuovi valori di Start, Stop e Step possono essere impostati, ed il parametro di input può anche essere completamente escluso dall'ottimizzazione indipendentemente dalle impostazioni Tester della strategia. Ciò consente di gestire l'area dei parametri di input durante l'ottimizzazione escludendo alcuni parametri dall'ottimizzazione in base ai valori dei parametri chiave dell' Expert Advisor".

Esempio:

```
#property description "Expert Advisor per la dimostrazione della funzione ParameterGet
#property description "Dev'essere lanciato in modalità ottimizzazione dello Strategy T
//--- parametri di input
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                                    |
//+------------------------------------------------------------------------------+
int OnInit()
  {
//--- L' Expert Advisor è designato per l'operazione solo nello the Strategy Tester
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Dev'essere lanciato in modalità ottimizzazione dello Strategy Teste
      //--- finisce l'operazione dell' Expert Advisor in anticipo e lo rimuove dal cha
      return(INIT_FAILED);
     }
//--- completamento dell'inizializzazione, con successo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------+
//| Funzione TesterInit                                                          |
//+------------------------------------------------------------------------------+
void OnTesterInit()
  {
//--- esempio per il parametro di input di tipo long
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
```

```
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("Primo parametro");
   PrintFormat("%s=%d  abilita=%s  da %d to %d con lo step=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- esempio di parametro di input per il tipo double
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("Secondo parametro");
   PrintFormat("%s=%G  abilita=%s  da %G to %G con lo step=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- esempio per il parametro di input di tipo bool
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("Terzo parametro");
   PrintFormat("%s=%s  abilita=%s  da %s to %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- esempio per il parametro di input di tipo enumerazione
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("Quarto parametro");
   PrintFormat("%s=%s  abilita=%s  da %s to %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
  }
//+------------------------------------------------------------------------------+
//| Funzione TesterDeinit                                                        |
//+------------------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- questo messaggio verrà mostrato dopo che l'ottimizzazzione è completata
   Print(__FUNCTION__," Ottimizzazione completata");
  }
```

ParameterSetRange

Specifica l'uso dei parametri di input quando si ottimizza un Expert Advisor nello Strategy Tester: valore, step del cambiamento, i valori iniziali e finali. Ci sono due varianti della funzione.

1. Specifica i valori per il parametro di input di tipo integer

```
bool  ParameterSetRange(
   const string  name,          // nome parametro (input variable)
   bool          enable,        // ottimizzaziona parametri abilitata
   long          value,         // valori parametro
   long          start,         // valore iniziale
   long          step,          // step del cambiamento
   long          stop           // valore finale
   );
```

2. Specifica i valori per il parametro di input di tipo real

```
bool  ParameterSetRange(
   const string  name,          // nome parametro (input variable)
   bool          enable,        // ottimizzaziona parametri abilitata
   double        value,         // valori parametro
   double        start,         // valore iniziale
   double        step,          // step del cambiamento
   double        stop           // valore finale
   );
```

Parametri

name

[in] input or sinput ID della variabile. Queste variabili sono parametri esterni di un'applicazione. I loro valori possono essere specificati al momento del lancio del programma.

enable

[in] Abilitare questo parametro per enumerare i valori durante l'ottimizzazione nello StrategyTester.

valore

[in] Valore parametro.

start

[in] il valore del parametro iniziale durante l'ottimizzazione.

step

[in] Cambiamento dello step parametri durante l'enumerazione dei suoi valori.

stop

[in] il valore del parametro finale durante l'ottimizzazione.

Valore restituito

Restituisce true se ha successo, altrimenti false. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

La funzione può essere chiamata solo dal gestore OnTesterInit() al momento del lancio dell' ottimizzazione dallo Strategy Tester. È stato designato per specificare il range dei parametri e lo step del cambiamento. Il parametro può essere completamente escluso dall' ottimizzazione indipendentemente dalle impostazioni dello Strategy Tester. Consente inoltre di utilizzare le variabili dichiarate con modificatore sinput nel processo di ottimizzazione.

ParameterSetRange() consente di gestire l' ottimizzazione di un Expert Advisor nello Strategy Tester a seconda dei valori dei suoi parametri chiave, includendo o escludendo i parametri di input richiesti dal l'ottimizzazione ed impostando il range richiesto e lo step del cambiamento.

Funzioni Eventi

Questo gruppo contiene le funzioni per lavorare con gli eventi personalizzati e le funzioni timer. Oltre a questo gruppo, ci sono funzioni speciali per l' handling di eventi predefiniti.

Funzione	Azione
EventSetMillisecondTimer	Avvia il generatore evento del timer ad alta risoluzione con un periodo di meno di 1 secondo per il chart corrente
EventSetTimer	Avvia il generatore di eventi timer con la periodicità specificata per il grafico corrente
EventKillTimer	Interrompe la generazione di eventi dal timer nel grafico corrente
EventChartCustom	Genera un evento personalizzato per il grafico specificato

Vedi anche

Tipi di Eventi del Grafico

EventSetMillisecondTimer

La funzione indica al terminale client che gli eventi timer dovrebbero essere generati ad intervalli di meno di un secondo, per questo Expert Advisor o indicatore.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // numero di millisecondi
   );
```

Parametri

millisecondi

[in] Numero di millisecondi che definiscono la frequenza degli eventi timer.

Valore restituito

In caso di esecuzione di successo, restituisce true, in caso contrario - false. Per ricevere un codice d'errore, dev'essere chiamata la funzione GetLastError().

Nota

Questa funzione è progettata per i casi in cui è necessario il timer ad alta risoluzione. In altre parole, gli eventi timer devono essere ricevuti più di una volta al secondo. Se un timer convenzionale con il periodo di più di un secondo è abbastanza per voi, usate EventSetTimer().

In generale, quando il periodo del timer viene ridotto, il tempo di prova è aumentato, come gestore di eventi timer viene chiamato più spesso. Quando si lavora in modalità real-time, gli eventi timer vengono generati non più di 1 volta a 10-16 millisecondi a causa di limitazioni hardware.

Di solito, questa funzione dovrebbe essere chiamata dalla funzione OnInit() o nella classe costruttore. Per gestire gli eventi provenienti dal timer, un Expert Advisor o un indicatore dovrebbe avere la funzione OnTimer().

Ogni Expert Advisor e ogni indicatore lavorano con un proprio timer ricevendo eventi esclusivamente da questo timer. Durante la chiusura dell'applicazione mql5, il timer viene forzatamente distrutto nel caso in cui è stato creato, ma non è stato disabilitato dala funzione EventKillTimer().

Solo un timer può essere lanciato per ogni programma. Ogni applicazione mql5 e chart hanno la loro coda di eventi in cui vengono inseriti tutti gli eventi appena arrivati. Se la coda contiene già l'evento Timer o questo evento è in fase di lavorazione, il nuovo evento Timer non viene aggiunto alla coda dell' applicazione mql5.

EventSetTimer

La funzione indica al terminale client, che per questo indicatore o Expert Advisor, gli eventi dal timer devono essere generati con la periodicità specificata.

```
bool  EventSetTimer(
   int  seconds      // numero di secondi
   );
```

Parametri

seconds

[in] Numero di secondi che determinano la frequenza del verificarsi di un evento timer.

Valore restituito

In caso di successo restituisce true, altrimenti false. Al fine di ottenere un codice di errore, deve essere chiamata la funzione GetLastError().

Nota

Normalmente, questa funzione deve essere chiamata dalla funzione OnInit() o da una classe costruttore. Al fine di gestire gli eventi provenienti dal timer, l'Expert Advisor deve avere la funzione OnTimer().

Ogni Expert Advisor, così come ogni Indicatore lavora con il proprio timer e riceve gli eventi solo da esso. Non appena un programma MQL5 si arresta, il timer viene distrutto forzatamente se è stato creato ma non è stato disabilitato dalla funzione EventKillTimer().

Per ciascun programma non più di un timer può essere eseguito. Ogni programma MQL5 ed ogni grafico ha la propria coda di eventi, in cui si trovano tutti gli eventi appena ricevuti. Se l'evento Timer è presente nella coda o viene elaborato, il nuovo evento Timer non verrà inserito nella coda del programma MQL5.

EventKillTimer

Specifica il terminale client che è necessario fermare la generazione di eventi da Timer.

```
void  EventKillTimer();
```

Valore restituito

Nessun valore restituito.

Nota

In genere, questa funzione deve essere richiamata da una funzione OnDeinit(), se la funzione EventSetTimer() è stata richiamata da OnInit(). Questa funzione può anche essere chiamata dal distruttore della classe, se la funzione EventSetTimer () è stata chiamata nel costruttore di questa classe.

Ogni Expert Advisor, così come ogni Indicatore lavora con il proprio timer e riceve gli eventi solo da esso. Non appena un programma MQL5 si arresta, il timer viene distrutto forzatamente se è stato creato ma non è stato disabilitato dalla funzione EventKillTimer()

EventChartCustom

La funzione genera un evento personalizzato per il grafico specificato.

```
bool  EventChartCustom(
   long    chart_id,            // identificatore dell'evento ricezione grafico
   ushort  custom_event_id,     // identificatore evento
   long    lparam,              // parametro di tipo long
   double  dparam,              // parametro di tipo double
   string  sparam               // parametro di tipo stringa
   );
```

Parametri

chart_id

[in] Identificatore del grafico. 0 significa il grafico corrente.

custom_event_id

[in] ID dell'evento utente. Questo identificatore viene automaticamente aggiunto al valore CHARTEVENT_CUSTOM e convertito nel tipo intero.

lparam

[in] Il parametro di evento del tipo long passato alla funzione OnChartEvent.

dparam

[in] Il parametro di evento del tipo double passato alla funzione OnChartEvent.

sparam

[in] Il parametro di evento di tipo stringa passato alla funzione OnChartEvent. Se la stringa è più lunga di 63 caratteri, viene troncata.

Valore restituito

Restituisce true se un evento personalizzato è stato inserito con successo nella coda eventi del grafico che riceve gli eventi. In caso di fallimento, restituisce false. Utilizzare GetLastError() per ottenere un codice di errore.

Nota

Un Expert Advisor o indicatore attaccato al grafico specificato gestisce l'evento utilizzando la funzione OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam).

Per ciascun tipo di evento, i parametri di input della funzione OnChartEvent() hanno valori definiti che sono necessari per l'elaborazione di questo evento. Gli eventi ed i valori passati attraverso questi parametri sono elencati nella tabella seguente.

Evento	Valore del parametro id	Valore del parametro lparam	Valore del parametro dparam	Valore del parametro sparam
Evento di una sequenza di tasti	CHARTEVENT_KEYDOWN	codice di un tasto premuto	Ripete in conteggio (il numero di volte	Il valore di stringa di una maschera di bit

Evento	Valore del parametro id	Valore del parametro lparam	Valore del parametro dparam	Valore del parametro sparam
			che il tasto viene ripetuto come risultato dell'utente che tiene premuto il tasto)	che descrive lo status dei pulsanti della tastiera
Evento mouse (se proprietà CHART_EVENT_MOUSE_MOVE=true viene impostato per il grafico)	CHARTEVENT_MOUSE_MOVE	la coordinata X	la coordinata Y	Il valore di stringa di una maschera di bit che descrive lo stato dei pulsanti del mouse
Evento di creazione di oggetti grafici (Se CHART_EVENT_OBJECT_CREATE=true è impostato per il grafico)	CHARTEVENT_OBJECT_CREATE	—	—	Nome dell'oggetto grafico creato
Evento di cambiamento di proprietà di un oggetto attraverso la finestra delle proprietà	CHARTEVENT_OBJECT_CHANGE	—	—	Nome dell'oggetto grafico modificato
Evento di eliminazione oggetto grafico (se CHART_EVENT_OBJECT_DELETE= true è impostato per il grafico)	CHARTEVENT_OBJECT_DELETE	—	—	Nome dell'oggetto grafico eliminato
Evento di un click del mouse sul grafico	CHARTEVENT_CLICK	la coordinata X	la coordinata Y	—
Evento di un clic del mouse in un oggetto grafico	CHARTEVENT_OBJECT_CLICK	la coordinata X	la coordinata Y	Nome dell'oggetto grafico, in cui

Evento	Valore del parametro id	Valore del parametro lparam	Valore del parametro dparam	Valore del parametro sparam
appartenente alla tabella				l'evento si è verificato
Evento di trascinamento di un oggetto grafico con il mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome dell'oggetto grafico spostato
Evento del testo finito di modifica nella casella di immissione dell'oggetto grafico LabelEdit	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome dell'oggetto grafico LabelEdit, in cui la modifica del testo è stata completata
Evento di modifiche in un grafico	CHARTEVENT_CHART_CHANGE	—	—	—
ID dell'evento dell'utente con il numero N	CHARTEVENT_CUSTOM+N	Valore impostato dalla funzione EventChartCustom()	Valore impostato dalla funzione EventChartCustom()	Valore impostato dalla funzione EventChartCustom()

Esempio:

```
//+------------------------------------------------------------------+
//|                                                ButtonClickExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
string labelID="Info";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| Funzione di inizializzazione dell' Expert                        |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Crea un bottone per inviare gli eventi personalizzati
   ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
   ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
```

```
   ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
   ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
   ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
   ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
   ObjectSetString(0,buttonID,OBJPROP_TEXT,"Button");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- Crea un'etichetta per visualizzare le informazioni
   ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
   ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
   ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
   ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
   ObjectSetString(0,labelID,OBJPROP_TEXT,"No information");
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------------------------+
//| Funzione deinizializzazione Expert                                                 |
//+------------------------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------------------------+
//| Funzione tick dell'Expert                                                          |
//+------------------------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Controlla l'evento premendo un pulsante del mouse
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
```

```
      string clickedChartObject=sparam;
      //--- Se si fa clic sull'oggetto con il nome buttonID
      if(clickedChartObject==buttonID)
        {
         //--- Stato del tasto - premuto o no
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- registra un messaggio di debug
         Print("Bottone premuto = ",selected);
         int customEventID; // Numero di evento custom da inviare
         string message;    // Messaggio da inviare nell'evento
         //--- Se il tasto viene premuto
         if(selected)
           {
            message="Tasto premuto";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // Il tasto non è premuto
           {
            message="Il tasto non è premuto";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- Invia un evento custom al "nostro" grafico
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         ///--- Invia un messaggio per aprire tutti i charts
         BroadcastEvent(ChartID(),0,"Broadcast Message");
         //--- Messaggio di debug
         Print("Invia un evento con ID = ",customEventID);
        }
      ChartRedraw();// Ridisegno forzato di tutti gli oggetti del grafico
     }

//--- Controlla l'evento che riguarda gli eventi utente
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Ottiene il messaggio trasmesso da una tabella con id = "+lparam);
        }
      else
        {
         //--- Leggiamo un messaggio di testo nell'event
         string info=sparam;
         Print("Gestisce l'evento utente con l' ID = ",id);
         //--- Mostra un messaggio nell'etichetta
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// Ridisegna forzatamente tutti gli oggetti del grafico
        }
     }
  }
```

```
//+------------------------------------------------------------------------------+
//| invia l'evento trasmesso a tutti i grafici aperti                            |
//+------------------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)                 // Abbiamo di certo non più di CHARTS_MAX grafi
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // Abbiamo ricevuto un nuovo grafico dal precede
      if(currChart==-1) break;        // Raggiunta la fine della lista dei grafici
      i++;// Non dimentichiamo di incrementare il contatore
     }
  }
//+------------------------------------------------------------------------------+
```

Vedi anche

Eventi del terminale client, Funzioni di gestione degli eventi

Lavorare con OpenCL

I programmi in OpenCL sono utilizzati per l'esecuzione di calcoli sulle schede video che supportano OpenCL 1.1 o superiore. Schede video moderne contengono centinaia di piccoli processori specializzati che possono contemporaneamente eseguire semplici operazioni matematiche con i flussi di dati in entrata. Il linguaggio OpenCL organizza il calcolo parallelo ed offre una maggiore velocità per una certa classe di tasks.

In alcune schede grafiche il lavoro con i tipi di numeri double è disabilitato di default. Questo può portare ad un errore di compilazione 5105. Per abilitare il supporto per i numeri di tipo double, si prega di aggiungere la seguente direttiva al programma OpenCL: #pragma OPENCL EXTENSION cl_khr_fp64 : enable. Tuttavia, se una scheda grafica non supporta i double, l'abilitazione di questa direttiva non sarà di aiuto.

Si raccomanda di scrivere il codice sorgente per OpenCL in file separati CL, che possono essere successivamente inseriti nel programma MQL5 utilizzando le variabili di risorse.

Gestione degli errori nei programmi OpenCL

Per ottenere informazioni sull'ultimo errore in un programma OpenCL, utilizzare le funzioniCLGetInfoIntegereCLGetInfoStringche permettono di ottenere il codice dell'errore e la descrizione del testo.

OpenCL ultimo codice errore : Per ottenere l'ultimo errore OpenCL, chiamareCLGetInfoInteger, mentre il parametro*handle*viene ignorato (può essere impostato a zero). Descrizione degli errori:https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS.

Per un codice errore sconosciuto, viene restituita la stringa "errore OpenCL sconosciuto N" dove N è un codice di errore. Esempio:

```
//--- il primo parametro 'handle' viene ignorato quando si ottiene l'ultimo codice er
intcode=(int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

Descrizione testuale dell'errore OpenCL: Per ottenere l'ultimo errore OpenCL, chiamareCLGetInfoString. Il codice di errore deve essere passato tramite il parametrohandle.

Descrizione degli errori:>https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Se viene passato CL_LAST_ERROR invece del codice di errore, la funzione restituisce la descrizione dell'ultimo errore. Per esempio:

```
//--- ottenere il codice dell'ultimo errore OpenCL
int   code= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
stringdesc;// per ottenere una descrizione di testo dell'errore

//--- usa il codice dell'errore per ottenere una descrizione di testo dell'errore
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
 desc="impossibile ottenere la descrizione dell'errore OpenCL,"+ (string)GetLastError
Print(desc);

//--- per ottenere la descrizione dell'ultimo errore OpenCL senza prima ottenere il c
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION,desc))
 desc="impossibile ottenere la descrizione dell'errore OpenCL,"+ (string)GetLastError
```

```
Print(desc);;
```

Finora, il nome dell'enumerazione interna è dato come descrizione di un errore. Potete trovare la sua codifica qui:https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Per esempio, il valore CL_INVALID_KERNEL_ARGS significa "Restituito durante l'accodamento di un kernel quando alcuni argomenti del kernel non sono stati impostati o non sono validi."

Funzioni per l'esecuzione di programmi in OpenCL:

Funzione	Azione
CLHandleType	Restituisce il tipo di un handle OpenCL come valore dell'enumerazione ENUM_OPENCL_HANDLE_TYPE
CLGetInfoInteger	Restituisce il valore di una proprietà integer per un oggetto OpenCL o dispositivo
CLContextCreate	Crea un contesto OpenCL
CLContextFree	Rimuove un contesto OpenCL
CLGetDeviceInfo	Riceve la proprietà del dispositivo dal driver di OpenCL
CLProgramCreate	Crea un programma di OpenCL da un codice sorgente
CLProgramFree	Rimuove un programma di OpenCL
CLKernelCreate	Crea una funzione di avvio OpenCL
CLKernelFree	Rimuove una funzione di avvio OpenCL
CLSetKernelArg	Imposta un parametro per la funzione OpenCL
CLSetKernelArgMem	Imposta un buffer OpenCL come parametro della funzione OpenCL
CLSetKernelArgMemLocal	Imposta il buffer locale come argomento della funzione del kernel
CLBufferCreate	Crea un buffer di OpenCL
CLBufferFree	Elimina un buffer OpenCL
CLBufferWrite	Scrive un array in un buffer OpenCL
CLBufferRead	Legge un buffer OpenCL in un array
CLExecute	Esegue un programma OpenCL
CLExecutionStatus	Restituisce lo stato di esecuzione del programma OpenCL

Vedi anche

OpenCL, Risorse

CLHandleType

Restituisce il tipo di un handle OpenCL come valore dell'enumerazione ENUM_OPENCL_HANDLE_TYPE.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // Handle di un oggetto OpenCL
   );
```

Parametri

handle

[in] Un handle per un oggetto OpenCL: un contesto, un kernel o un programma OpenCL.

Valore restituito

Il tipo di handle OpenCL come valore dell'enumerazione ENUM_OPENCL_HANDLE_TYPE.

ENUM_OPENCL_HANDLE_TYPE

Identificatore	Descrizione
OPENCL_INVALID	Incorrect handle
OPENCL_CONTEXT	Un handle del contesto OpenCL
OPENCL_PROGRAM	Un handle del programma OpenCL
OPENCL_KERNEL	Un handle del kernel OpenCL
OPENCL_BUFFER	Un handle del buffer OpenCL

CLGetInfoInteger

Restituisce il valore di una proprietà integer per un oggetto o dispositivo OpenCL.

```
long  CLGetInfoInteger(
   int  handle,                          // L' handle dell' oggetto OpenCL o il numer
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // Proprietà richiesta
   );
```

Parametri

handle

[in] Un handle per l'oggetto OpenCL o il numero del dispositivo OpenCL. La numerazione dei dispositivi OpenCL inizia con zero.

prop

[in] Il tipo di una proprietà richiesta dall' enumarazione ENUM_OPENCL_PROPERTY_INTEGER, il cui valore si desidera ottenere.

Valore restituito

Il valore della proprietà in caso di successo o -1 in caso di errore. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

ENUM_OPENCL_PROPERTY_INTEGER

Identificatore	Descrizione	Tipo
CL_DEVICE_COUNT	Il numero di dispositivi con supporto OpenCL. Questa proprietà non richiede specifiche del primo parametro, cioè si può passare un valore zero per il parametro *handle*.	int
CL_DEVICE_TYPE	Tipo di dispositivo	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	Identificatore unico del fornitore	uint
CL_DEVICE_MAX_COMPUTE_UNITS	Numero di attività parallele calcolate nel dispositivo OpenCL. Un gruppo di lavoro risolve un compito computazionale. Il valore minimo è 1	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	Set massima frequenza del dispositivo in MHz.	uint
CL_DEVICE_GLOBAL_MEM_SIZE	Dimensione della memoria globale del dispositivo in byte	ulong

Identificatore	Descrizione	Tipo
CL_DEVICE_LOCAL_MEM_SIZE	Dimensione dei dati trattati (scena) di memoria locale in byte	uint
CL_BUFFER_SIZE	Grandezza effettiva del buffer OpenCL in byte	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	Il numero totale dei gruppi di lavoro locali disponibili per un dispositivo OpenCL.	ulong
CL_KERNEL_WORK_GROUP_SIZE	Il numero totale dei gruppi di lavoro locali disponibili per un programma OpenCL.	ulong
CL_KERNEL_LOCAL_MEM_SIZE	Grandezza della memoria locale (in byte) utilizzata da un programma OpenCL per risolvere tutti i task paralleli in un gruppo. Utilizzare CL_DEVICE_LOCAL_MEM_SIZE per ricevere il valore massimo disponibile	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	La grandezza minima della memoria privata (in byte) utilizzata da ogni attività nel kernel del programma OpenCL.	ulong
CL_LAST_ERROR	Il valore dell'ultimo errore OpenCL	int

L'enumerazione ENUM_CL_DEVICE_TYPE contiene i possibili tipi di dispositivi che supportano OpenCL. È possibile ricevere il tipo di dispositivo con il suo numero o l'handle dell'oggetto OpenCL chiamando CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE).

ENUM_CL_DEVICE_TYPE

Identificatore	Descrizione
CL_DEVICE_ACCELERATOR	Acceleratori dedicati OpenCL (per esempio, the IBM CELL Blade). Questi dispositivi comunicano con il processore host usando una interconnessione periferica come PCIe.
CL_DEVICE_CPU	Un dispositivo OpenCL è il processore host. Il processore host esegue le implementazioni OpenCL ed è una CPU singola o CPU multi-core.
CL_DEVICE_GPU	Un dispositivo OpenCL che è una GPU.

Identificatore	Descrizione
CL_DEVICE_DEFAULT	Il dispositivo OpenCL predefinito nel sistema. Il dispositivo predefinito non può essere un dispositivo CL_DEVICE_TYPE_CUSTOM.
CL_DEVICE_CUSTOM	Acceleratori dedicati che non supportano i programmi scritti in OpenCL C.

Esempio:

```
voidOnStart()
  {
   int cl_ctx;
//--- inizializza il contesto OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL non trovato");
      return;
     }
//--- Mostra informazioni generiche riguardo il dispositivo OpenCL
   Print("Tipo OpenCL: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY),"  MHz"
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE),"  by
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE),"  byte
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

CLGetInfoString

Restituisce il valore stringa di una proprietà per l'oggetto o dispositivo OpenCL.

```
bool  CLGetInfoString(
   int  handle,                          // Handle di oggetto OpenCL o numero del dis
   ENUM_OPENCL_PROPERTY_STRING  prop,    // proprietà richiesta
   string&  value                        // stringa referenziata
   );
```

Parametri

handle

[in] andle di oggetto OpenCL o numero di dispositivo OpenCL. La numerazione dei dispositivi OpenCL inizia con zero.

prop

[in] Tipo della proprietà richiesta, dall'enumerazione ENUM_OPENCL_PROPERTY_STRING, il cui valore deve essere ottenuto.

valore

[out] Stringa per ricevere il valore della proprietà.

Valore restituito

vero se ha successo, altrimenti false. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

ENUM_OPENCL_PROPERTY_STRING

Identificatore	Descrizione
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - Profilo OpenCL. Nome del profilo può essere uno dei seguenti valori: • FULL_PROFILE - implementazione supporto OpenCL (la funzionalità è definita come la parte della specifica del kernel senza richiedere ulteriori estensioni per il supporto OpenCL); • EMBEDDED_PROFILE - implementazione supporto OpenCL come supplemento. Il profilo modificato è definito come un sottoinsieme per ogni versione OpenCL.
CL_PLATFORM_VERSION	Versione OpenCL
CL_PLATFORM_VENDOR	Nome venditore piattaforma
CL_PLATFORM_EXTENSIONS	Elenco delle estensioni supportate dalla piattaforma. I nomi di estensione devono essere supportati da tutti i dispositivi correlati a questa piattaforma
CL_DEVICE_NAME	Nome dispositivo

Identificatore	Descrizione
CL_DEVICE_VENDOR	Nome del venditore
CL_DRIVER_VERSION	Versione del driver OpenCL in formato major_number.minor_number
CL_DEVICE_PROFILE	Dispositivo profilo OpenCL. Il nome del profilo può essere uno dei seguenti valori: • FULL_PROFILE - implementazione supporto OpenCL (la funzionalità è definita come la parte della specifica del kernel senza richiedere ulteriori estensioni per il supporto OpenCL); • EMBEDDED_PROFILE - implementazione supporto OpenCL come supplemento. Il profilo modificato è definito come un sottoinsieme per ogni versione OpenCL.
CL_DEVICE_VERSION	Versione OpenCL in formato "OpenCL <space><major_version.minor_version><space><vendor-specific information>"
CL_DEVICE_EXTENSIONS	Elenco delle estensioni supportate dal dispositivo. L'elenco può contenere estensioni supportate dal venditore, nonché uno o più nomi approvati: cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing
CL_DEVICE_BUILT_IN_KERNELS	L'elenco dei kernel supportati separati da ";".
CL_DEVICE_OPENCL_C_VERSION	La versione massima supportata dal compilatore per questo dispositivo. Versione del formato: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "
CL_ERROR_DESCRIPTION	Descrizione testuale di un errore OpenCL

Esempio:

```
voidOnStart()
  {
   int cl_ctx;
   string str;
//--- inizializza il contesto OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
```

```
      Print("OpenCL non trovato");
      return;
     }
//--- Mostra le informazioni riguardo la piattaforma
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("Nome piattaforma OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("Venditore piattaforma OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("Versione piattaforma OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("Profilo piattaforma OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("Ext(estensione) piattaforma OpenCL: ",str);
//--- Mostra le informazioni riguardo il dispositivo
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("nome del dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("profilo del dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("kernel del dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("ext del dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("Venditore del dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("Versione del dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c ver: ",str);
//--- Mostra informazioni generiche riguardo il dispositivo OpenCL
   Print("Tipo OpenCL: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

CLContextCreate

Crea un contesto di OpenCL e restituisce il suo handle.

```
int  CLContextCreate(
   int  device=CL_USE_ANY     // Numero seriale di dispositivi OpenCL o macro
   );
```

Parametro

dispositivo

[in] Il numero ordinale del dispositivo OpenCL nel sistema. Invece di un numero specifico, è possibile specificare uno dei seguenti valori:

- CL_USE_ANY - un qualsiasi dispositivo disponibile con supporto OpenCL, è consentito;
- CL_USE_CPU_ONLY - solo emulazione OpenCL su CPU è consentita;
- CL_USE_GPU_ONLY - l' emulazione OpenCL è proibita e solo dispositivi specializzati con supporto OpenCL (schede video) possono essere usati;
- CL_USE_GPU_DOUBLE_ONLY - sono consentite solo le GPU che supportano il tipo double.

Valore restituito

Un handle per il contesto OpenCL in caso di successo, altrimenti -1. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

CLContextFree

Rimuove un contesto OpenCL.

```
void  CLContextFree(
   int  context     // Handle ad un contesto OpenCL
   );
```

Parametri

contesto

[in] Handle del contesto OpenCL.

Valore restituito

Nessuno. Nel caso di un errore interno il valore _LastError cambia. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

CLGetDeviceInfo

La funzione riceve la proprietà del dispositivo dal driver OpenCL.

```
bool  CLGetDeviceInfo(
   int     handle,          // Hande del dispositivo OpenCL
   int     property_id,     // proprietà ID richiesta
   uchar&  data[],          // array per la ricezione dati
   uint&   size             // slitta gli elementi array, il valore di default è 0
   );
```

Parametri

handle

[in] L'Indice del dispositivo OpenCL o l'handle OpenCL creato dalla funzione CLContextCreate().

property_id

[in] ID della proprietà del dispositivo OpenCL che dev'essere ricevuta. I valori possono essere di uno di quelli predeterminati elencati nella tabella sottostante.

data[]

[out] L'array per la ricezione di dati sulla proprietà richiesta.

grandezza

[out] Grandezza dei dati ricevuti nell'array *dati[]*.

Valore restituito

vero se ha successo, altrimenti false. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

Per array uni-dimensionali, il numero dell'elemento, da cui partono i dati per la lettura di Buffer OpenCL, è calcolato tenendo conto della flag AS_SERIES.

L'elenco di ID disponibili sulle proprietà del dispositivo OpenCL

La descrizione esatta della proprietà e le sue funzioni possono essere trovate all'indirizzo del sito web ufficiale OpenCL.

Identificatore	Valore
CL_DEVICE_TYPE	0x1000
CL_DEVICE_VENDOR_ID	0x1001
CL_DEVICE_MAX_COMPUTE_UNITS	0x1002
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS	0x1003
CL_DEVICE_MAX_WORK_GROUP_SIZE	0x1004
CL_DEVICE_MAX_WORK_ITEM_SIZES	0x1005

Identificatore	Valore
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT	0x1007
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT	0x1008
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG	0x1009
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT	0x100A
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE	0x100B
CL_DEVICE_MAX_CLOCK_FREQUENCY	0x100C
CL_DEVICE_ADDRESS_BITS	0x100D
CL_DEVICE_MAX_READ_IMAGE_ARGS	0x100E
CL_DEVICE_MAX_WRITE_IMAGE_ARGS	0x100F
CL_DEVICE_MAX_MEM_ALLOC_SIZE	0x1010
CL_DEVICE_IMAGE2D_MAX_WIDTH	0x1011
CL_DEVICE_IMAGE2D_MAX_HEIGHT	0x1012
CL_DEVICE_IMAGE3D_MAX_WIDTH	0x1013
CL_DEVICE_IMAGE3D_MAX_HEIGHT	0x1014
CL_DEVICE_IMAGE3D_MAX_DEPTH	0x1015
CL_DEVICE_IMAGE_SUPPORT	0x1016
CL_DEVICE_MAX_PARAMETER_SIZE	0x1017
CL_DEVICE_MAX_SAMPLERS	0x1018
CL_DEVICE_MEM_BASE_ADDR_ALIGN	0x1019
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE	0x101A
CL_DEVICE_SINGLE_FP_CONFIG	0x101B
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE	0x101C
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE	0x101D
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE	0x101E
CL_DEVICE_GLOBAL_MEM_SIZE	0x101F
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE	0x1020

Identificatore	Valore
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022
CL_DEVICE_LOCAL_MEM_SIZE	0x1023
CL_DEVICE_ERROR_CORRECTION_SUPPORT	0x1024
CL_DEVICE_PROFILING_TIMER_RESOLUTION	0x1025
CL_DEVICE_ENDIAN_LITTLE	0x1026
CL_DEVICE_AVAILABLE	0x1027
CL_DEVICE_COMPILER_AVAILABLE	0x1028
CL_DEVICE_EXECUTION_CAPABILITIES	0x1029
CL_DEVICE_QUEUE_PROPERTIES	0x102A
CL_DEVICE_NAME	0x102B
CL_DEVICE_VENDOR	0x102C
CL_DRIVER_VERSION	0x102D
CL_DEVICE_PROFILE	0x102E
CL_DEVICE_VERSION	0x102F
CL_DEVICE_EXTENSIONS	0x1030
CL_DEVICE_PLATFORM	0x1031
CL_DEVICE_DOUBLE_FP_CONFIG	0x1032
CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF	0x1034
CL_DEVICE_HOST_UNIFIED_MEMORY	0x1035
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR	0x1036
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT	0x1037
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT	0x1038
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG	0x1039
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	0x103A
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	0x103B
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0x103C
CL_DEVICE_OPENCL_C_VERSION	0x103D
CL_DEVICE_LINKER_AVAILABLE	0x103E
CL_DEVICE_BUILT_IN_KERNELS	0x103F

Identificatore	Valore
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041
CL_DEVICE_PARENT_DEVICE	0x1042
CL_DEVICE_PARTITION_MAX_SUB_DEVICES	0x1043
CL_DEVICE_PARTITION_PROPERTIES	0x1044
CL_DEVICE_PARTITION_AFFINITY_DOMAIN	0x1045
CL_DEVICE_PARTITION_TYPE	0x1046
CL_DEVICE_REFERENCE_COUNT	0x1047
CL_DEVICE_PREFERRED_INTEROP_USER_SYNC	0x1048
CL_DEVICE_PRINTF_BUFFER_SIZE	0x1049
CL_DEVICE_IMAGE_PITCH_ALIGNMENT	0x104A
CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMENT	0x104B

Esempio:

```
voidOnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERRORE in CLContextCreate");
      string device;
      CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
      Print(i,": ",device);
      uchar data[1024];
      uint size;
      CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
      Print("size = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- esempio di scritte nell' Expert Journal
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      2: Advanced Micro Devices, Inc.
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 32
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Tahiti
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Intel(R) Corporation
```

```
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 21
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      1:        Intel(R) Core(TM) i7-37
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

CLProgramCreate

Crea un programma OpenCL da un codice sorgente.

```
int  CLProgramCreate(
   int           context,     // Handle ad un contesto OpenCL
   const string  source       // Codice sorgente
   );
```

Parametri

contesto

[in] Handle del contesto OpenCL.

sorgente

[in] Stringa con il codice sorgente del programma OpenCL.

Valore restituito

Un handle per un oggetto OpenCL in caso di successo. In caso di errore viene restituito -1. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

Al momento, vengono utilizzati i codici di errore seguenti:

- ERR_OPENCL_INVALID_HANDLE - handle non valido al contesto OpenCL.
- ERR_INVALID_PARAMETER - parametro stringa non valido.
- ERR_NOT_ENOUGH_MEMORY - memoria insufficiente per completare l'operazione.
- ERR_OPENCL_PROGRAM_CREATE - errore interno di OpenCL o errore di compilazione.

In some graphic cards working with the double type numbers is disabled by default. This can lead to compilation error 5105. To enable support for the double type numbers, please add the following directive to your OpenCL program: #pragma OPENCL EXTENSION cl_khr_fp64 : enable

Example:

```
//+------------------------------------------------------------------+
//| OpenCL kernel                                                    |
//+------------------------------------------------------------------+
const string
cl_src=
       //--- by default some GPU doesn't support doubles
       //--- cl_khr_fp64 directive is used to enable work with doubles
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- OpenCL kernel function
       "__kernel void Test_GPU(__global double *data,      \r\n"
       "                       const    int N,             \r\n"
       "                       const    int total_arrays)  \r\n"
       "  {                                                 \r\n"
       "   uint kernel_index=get_global_id(0);              \r\n"
       "   if (kernel_index>total_arrays) return;           \r\n"
       "   uint local_start_offset=kernel_index*N;          \r\n"
```

```
       "   for(int i=0; i<N; i++)                                \r\n"
       "     {                                                   \r\n"
       "       data[i+local_start_offset] *= 2.0;                \r\n"
       "     }                                                   \r\n"
       "  }                                                      \r\n";
//+------------------------------------------------------------------+
//| Test_CPU                                                         |
//+------------------------------------------------------------------+
bool Test_CPU(double &data[],const int N,const int id,const int total_arrays)
  {
//--- check array size
   if(ArraySize(data)==0) return(false);
//--- check array index
   if(id>total_arrays) return(false);
//--- calculate local offset for array with index id
   int local_start_offset=id*N;
//--- multiply elements by 2
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // size of the array
#define TOTAL_ARRAYS 5    // total arrays
//--- OpenCL handles
int cl_ctx;  // OpenCL context handle
int cl_prg;  // OpenCL program handle
int cl_krn;  // OpenCL kernel handle
int cl_mem;  // OpenCL buffer handle
//---
double DataArray1[]; // data array for CPU calculation
double DataArray2[]; // data array for GPU calculation
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- initialize OpenCL objects
//--- create OpenCL context
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL not found. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL program
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
     {
```

```
      CLContextFree(cl_ctx);
      Print("OpenCL program create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL kernel
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
     {
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL kernel create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL buffer
   if((cl_mem=CLBufferCreate(cl_ctx,ARRAY_SIZE*TOTAL_ARRAYS*sizeof(double),CL_MEM_REAI
     {
      CLKernelFree(cl_krn);
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL buffer create failed. Error=",GetLastError());
      return(1);
     }
//--- set OpenCL kernel constant parameters
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- prepare data arrays
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- fill arrays with data
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local start offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- prepare array with index j
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         //--- fill arrays with function MathCos(i+j);
         DataArray1[i+local_offset]=MathCos(i+j);
         DataArray2[i+local_offset]=MathCos(i+j);
        }
     };
//--- test CPU calculation
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculation of the array with index j
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- prepare CLExecute params
   uint  offset[]={0};
```

```
//--- global work size
   uint  work[]={TOTAL_ARRAYS};
//--- write data to OpenCL buffer
   CLBufferWrite(cl_mem,DataArray2);
//--- execute OpenCL kernel
   CLExecute(cl_krn,1,offset,work);
//--- read data from OpenCL buffer
   CLBufferRead(cl_mem,DataArray2);
//--- total error
   double total_error=0;
//--- compare results and calculate error
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- compare the results
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         double v1=DataArray1[i+local_offset];
         double v2=DataArray2[i+local_offset];
         double delta=MathAbs(v2-v1);
         total_error+=delta;
         //--- show first and last arrays
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("array %d of %d, element [%d]:  %f, %f, [error]=%f",j+1,TOTAL_
        }
     }
   PrintFormat("Total error: %f",total_error);
//--- delete OpenCL objects
//--- free OpenCL buffer
   CLBufferFree(cl_mem);
//--- free OpenCL kernel
   CLKernelFree(cl_krn);
//--- free OpenCL program
   CLProgramFree(cl_prg);
//--- free OpenCL context
   CLContextFree(cl_ctx);
//---
   return(0);
  }
```

CLProgramFree

Rimuove un programma OpenCL.

```
void  CLProgramFree(
   int  program     // Handle all'oggetto OpenCL
   );
```

Parametri

program

[in] Handle dell'oggetto OpenCL.

Valore restituito

Nessuno. Nel caso di un errore interno il valore _LastError cambia. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

CLKernelCreate

Crea il kernel del programma OpenCL e restituisce il relativo handle.

```
int  CLKernelCreate(
   int           program,        // Handle ad un oggetto OpenCL
   const string  kernel_name     // Nome del Kernel
   );
```

Parametri

program

[in] Handle ad un oggetto del programma OpenCL.

kernel_name

[in] Il nome della funzione kernel nel programma appropriato OpenCL, in cui ha inizio l'esecuzione.

Valore restituito

Un handle per un oggetto OpenCL in caso di successo. In caso di errore viene restituito -1. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

Al momento, vengono utilizzati i codici di errore seguenti:

- ERR_OPENCL_INVALID_HANDLE - handle non valido al *programma* OpenCL.
- ERR_INVALID_PARAMETER - parametro stringa non valido.
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - il nome del kernel contiene più di 127 caratteri.
- ERR_OPENCL_KERNEL_CREATE - è avvenuto un errore interno mentre si creava un oggetto OpenCL.

CLKernelFree

Rimuove una funzione start di OpenCL.

```
void  CLKernelFree(
   int  kernel     // Handle al kernel di un programma OpenCL
   );
```

Parametri

kernel_name

[in] Handle di un oggetto kernel.

Valore restituito

Nessuno. Nel caso di un errore interno il valore _LastError cambia. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

CLSetKernelArg

Imposta un parametro per la funzione OpenCL.

```
bool  CLSetKernelArg(
   int   kernel,         // Handle al kernel di un programma OpenCL
   uint  arg_index,      // Il numero di argomenti della funzione OpenCL
   void  arg_value       // Codice sorgente
   );
```

Parametri

kernel

[in] Handle al kernel di un programma OpenCL.

arg_index

[in] Il numero di l'argomento della funzione, la numerazione inizia da zero.

arg_value

[in] Il valore dell'argomento della funzione.

Valore restituito

Restituisce vero se ha successo, altrimenti restituisce false. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

Al momento, vengono utilizzati i codici di errore seguenti:

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE - handle non valido al kernel OpenCL.
- ERR_OPENCL_SET_KERNEL_PARAMETER - errore interno di OpenCL.

CLSetKernelArgMem

Imposta un buffer OpenCL come parametro della funzione OpenCL.

```
bool  CLSetKernelArgMem(
   int   kernel,           // Handle al kernel di un programma OpenCL
   uint  arg_index,        // Il numero di argomenti della funzione OpenCL
   int   cl_mem_handle     // Handle ad un buffer OpenCL
   );
```

Parametri

kernel

[in] Handle al kernel di un programma OpenCL.

arg_index

[in] Il numero di l'argomento della funzione, la numerazione inizia da zero.

cl_mem_handle

[in] Un handle per un buffer OpenCL.

Valore restituito

Restituisce vero se ha successo, altrimenti restituisce false. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

CLSetKernelArgMemLocal

Imposta il buffer locale come argomento della funzione del kernel.

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // handle al kernel di un programma OpenCL
   uint   arg_index,        // numero di argomenti funzione OpenCL
   ulong  local_mem_size    // grandezza buffer
   );
```

Parametri

kernel

[in] Handle al kernel del programma OpenCL.

arg_index

[in] Il numero di argomento della funzione, la numerazione inizia con lo zero.

local_mem_size

[in] Grandezza del buffer in byte.

Valore di Ritorno

Restituisce true in caso di successo, altrimenti restituisce false. Per informazioni sull'errore, utilizzare la funzione GetLastError().

CLBufferCreate

Crea un buffer OpenCL e restituisce il suo handle.

```
int  CLBufferCreate(
   int   context,     // Handle ad un contesto OpenCL
   uint  size,        // Grandezza Buffer
   uint  flags        // Combinazione di Flags che specifica le proprietà di un buffe
   );
```

Parametri

contesto

[in] Handle per un contesto OpenCL.

grandezza

[in] Grandezza del buffer in byte.

flags

[in] Le proprietà del buffer che vengono impostate utilizzando una combinazione di flag: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

Valore restituito

Un handle per un buffer OpenCL in caso di successo. In caso di errore viene restituito -1. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

Al momento, vengono utilizzati i codici di errore seguenti:

- ERR_OPENCL_INVALID_HANDLE - handle non valido al contesto OpenCL.
- ERR_NOT_ENOUGH_MEMORY - memoria insufficiente.
- ERR_OPENCL_BUFFER_CREATE - errore interno creazione buffer.

CLBufferFree

Cancella un buffer OpenCL.

```
void  CLBufferFree(
   int   buffer     // Handle ad un buffer OpenCL
   );
```

Parametri

buffer

[in] Un handle per un buffer OpenCL.

Valore restituito

Nessuno. Nel caso di un errore interno il valore _LastError cambia. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

CLBufferWrite

Scrive nel buffer OpenCL e restituisce il numero di elementi scritti.

```
uint  CLBufferWrite(
   int           buffer,                    // Un handle per un buffer OpenCL
   const void&   data[],                    // Un array di valori
   uint          buffer_offset=0,           // Un offset nel buffer OpenCL in bytes, 0
   uint          data_offset=0,             // Un offset negli elementi dell' array, 0
   uint          data_count=WHOLE_ARRAY     // Il numero di valori dall'array per la sc
   );
```

Ci sono anche versioni per la gestione di matrici e vettori.

Scrive i valori dalla matrice nel buffer e restituisce true in caso di esito positivo.

```
uint  CLBufferWrite(
   int           buffer,                    // un handle al buffer OpenCL
   uint          buffer_offset,             // uno scostamento nel buffer OpenCL in by
   matrix<T>     &mat                       /i valori della matrice per la scrittura r
   );
```

Scrive i valori dal vettore nel buffer e restituisce true in caso di esito positivo.

```
uint  CLBufferWrite(
   int           buffer,                    // un handle al buffer OpenCL
   uint          buffer_offset,             // uno scostamento nel buffer OpenCL in by
   vector<T>     &vec                       // i valori del vettore per la scrittura r
   );
```

Parametri

buffer

[in] Un handle di un buffer OpenCL.

data[]

[in] Un array di valori che dovrebbero essere scritti nel buffer OpenCL. Passato per riferimento.

buffer_offset

[in] Un offset nel buffer OpenCL in byte, da cui inizia la scrittura. Per default, la scrittura inizia dall'inizio del buffer.

data_offset

[in] L'indice del primo elemento dell'array, a partire dal quale i valori dell'array sono scritti nel buffer OpenCL. Per impostazione predefinita, vengono presi i valori sin dall'inizio dell'array.

data_count

[in] Il numero di valori che devono essere scritti. Tutti i valori dell'array, di default.

mat

[out] La matrice per la lettura dei dati dal buffer può essere uno qualsiasi dei tre tipi: matrix, matrixf o matrixc.

vec

[out] Il vettore per la lettura dei dati dal buffer può essere di uno qualsiasi dei tre tipi — vector, vectorf or vectorc.

Valore restituito

Il numero di elementi scritti. 0 viene restituito in caso di errore. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

true se una matrice o un vettore vengono gestiti correttamente, altrimenti false.

Nota

Per array uni-dimensionali, il numero dell'elemento, con cui la lettura dei dati di scrittura in un inizio di buffer OpenCL, viene calcolata tenendo conto delle flags AS_SERIES.

Un array di due o più dimensioni è presentata come uni-dimensionale. In questo caso, *data_offset* è il numero di elementi che devono essere saltati nella presentazione, non il numero di elementi nella prima dimensione.

Esempio di moltiplicazione di matrici utilizzando il metodo MatMul e calcolo parallelo in OpenCL

```
#define M       3000      // il numero di righe nella prima matrice
#define K       2000      //il numero di colonne nella prima matrice è uguale al numer
#define N       3000      //il numero di colonne nella seconda matrice

//+------------------------------------------------------------------+
const string clSrc=
  "#define N     "+IntegerToString(N)+"                              \r\n"
  "#define K     "+IntegerToString(K)+"                              \r\n"
  "                                                                 \r\n"
  "__kernel void matricesMul( __global float *in1,                  \r\n"
  "                           __global float *in2,                  \r\n"
  "                           __global float *out  )                \r\n"
  "{                                                                \r\n"
  "  int m = get_global_id( 0 );                                    \r\n"
  "  int n = get_global_id( 1 );                                    \r\n"
  "  float sum = 0.0;                                               \r\n"
  "  for( int k = 0; k < K; k ++ )                                  \r\n"
  "     sum += in1[ m * K + k ] * in2[ k * N + n ];                 \r\n"
  "  out[ m * N + n ] = sum;                                        \r\n"
  "}                                                                \r\n";
//+------------------------------------------------------------------+
//| Programma Script funzione start                                  |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- inizializza il generatore di numeri casuali
  MathSrand((int)TimeCurrent());
//---riempie le matrici di una certa dimensione con valori casuali
```

```
  matrixf mat1(M, K, MatrixRandom) ;     // prima matrice
  matrixf mat2(K, N, MatrixRandom);      // seconda matrice

//--- calcola il prodotto delle matrici usando il metodo naive
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// il risultato della moltiplicazione di d
  for(int m=0; m<M; m++)
    for(int k=0; k<K; k++)
      for(int n=0; n<N; n++)
        matrix_naive[m][n]+=mat1[m][k]*mat2[k][n];
  uint time_naive=GetTickCount()-start;

//--- calcola il prodotto delle matrici tramite MatMull
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- calcola il prodotto delle matrici in OpenCL
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;              // contesto handle
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found, leaving");
    return;
   }
  int cl_prg;              // handle del programma
  int cl_krn;              // handle del kernel
  int cl_mem_in1;          // primo (input) handle del buffer<
  int cl_mem_in2;          // secondo (input) handle del buffer<
  int cl_mem_out;          // terzo (input) handle del buffer<
//--- creo il programma e il kernel
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- creo tutti e tre i buffer per le tre matrici
  cl_mem_in1=CLBufferCreate(cl_ctx, M*K*sizeof(float), CL_MEM_READ_WRITE);
  cl_mem_in2=CLBufferCreate(cl_ctx, K*N*sizeof(float), CL_MEM_READ_WRITE);
//---terza matrice - output
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- imposta gli argomenti del kernel
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- scrive le matrici nei buffer del dispositivo
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- Ora d'Inizio di esecuzione del codice OpenCL
  start=GetTickCount();
//--- imposta i parametri dell'area di lavoro dell'attività ed esegue il programma Ope
```

```
  uint  offs[2] = {0, 0};
  uint works[2] = {M, N};
  start=GetTickCount();
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- calcola il risultato sulla matrice
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("[%d x %d] matrix read: ", matrix_opencl.Rows(), matrix_opencl.Cols())
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl failed. Error ",GetLastError())
  uint time_opencl=GetTickCount()-start;
  Print("Compare calculation time using each method");
  PrintFormat("Naive product time = %d ms",time_naive);
  PrintFormat("MatMul product time = %d ms",time_matmul);
  PrintFormat("OpenCl product time = %d ms",time_opencl);
//--- rilascia tutti i contesti OpenCL
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- confronta tra loro tutti i risultati delle matrici ottenuti
  Print("How many discrepancy errors are there between result matrices?");
  ulong errors=matrix_naive.Compare(matrix_matmul,(float)1e-12);
  Print("matrix_direct.Compare(matrix_matmul,1e-12)=",errors);
  errors=matrix_matmul.Compare(matrix_opencl,float(1e-12));
  Print("matrix_matmul.Compare(matrix_opencl,1e-12)=",errors);
/*
  Risultato:

   [3000 x 3000] lettura matrice:
  Confronta il tempo di calcolo con ciascun metodo
   Naive tempo prodotto = 54750 ms
   MatMul tempo prodotto = 4578 ms
   OpenCl tempo prodotto = 922 ms
  Quanti errori di discrepanza ci sono tra i risultati delle matrici?
   matrix_direct.Confronto(matrix_matmul,1e-12)=0
   matrix_matmul.Confronto(matrix_opencl,1e-12)=0
*/
 }
//+------------------------------------------------------------------+
//| Riempie la matrici con valori casuali                            |
//+------------------------------------------------------------------+
void MatrixRandom(matrixf& m)
 {
  for(ulong r=0; r<m.Rows(); r++)
   {
    for(ulong c=0; c<m.Cols(); c++)
     {
      m[r][c]=(float)((MathRand()-16383.5)/32767.);
     }
   }
 }
```

```
//+------------------------------------------------------------------+
//| Rilascia tutti i contesti OpenCL                                 |
//+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
 {
//--- elimina tutti i contesti creati da OpenCL in ordine inverso
  CLBufferFree(cl_mem_in1);
  CLBufferFree(cl_mem_in2);
  CLBufferFree(cl_mem_out);
  CLKernelFree(cl_krn);
  CLProgramFree(cl_prg);
  CLContextFree(cl_ctx);
 }
```

CLBufferRead

Legge un buffer OpenCL in un array e restituisce il numero di elementi letti.

```
uint  CLBufferRead(
   int           buffer,                    // Un handle per un buffer OpenCL
   const void&   data[],                    // Un array di valori
   uint          buffer_offset=0,           // Un offset nel buffer OpenCL in bytes, 0
   uint          data_offset=0,             // Un offset negli elementi dell' array, 0
   uint          data_count=WHOLE_ARRAY     // Il numero di valori dal buffer per la le
   );
```

Ci sono anche versioni per la gestione di matrici e vettori.

Legge il buffer OpenCL nella matrice e restituisce true in caso di esito positivo.

```
uint  CLBufferRead(
   int           buffer,                    // un handle al buffer OpenCL
   uint          buffer_offset,             // uno scostamento nel buffer OpenCL in by
   const matrix& mat,                       // la matrice per ricevere i valori dal bu
   ulong         rows=-1,                   // il numero di righe nella matrice
   ulong         cols=-1                    // il numero di colonne nella matrice
   );
```

Legge il buffer OpenCL nel vettore e restituisce true in caso di esito positivo.

```
uint  CLBufferRead(
   int           buffer,                    // un handle al buffer OpenCL
   uint          buffer_offset,             // uno scostamento nel buffer OpenCL in by
   const vector& vec,                       // il vettore per ricevere i valori dal bu
   ulong         size-1,                    // lunghezza del vettore
   );
```

Parametri

buffer

[in] Un handle del buffer OpenCL .

data[]

[in] Un array per ricevere valori dal buffer OpenCL. Passato per riferimento.

buffer_offset

[in] Un offset nel buffer OpenCL in byte, da cui inizia la lettura. Per default, la lettura inizia proprio dall'inizio inizio, del buffer.

data_offset

[in] L'indice del primo elemento della matrice per scrivere i valori del buffer OpenCL. Per default, la scrittura dei valori letti in un array inizia dall'indice zero.

data_count

[in] Il numero di valori che devono essere letti. L'intero buffer OpenCL viene letto per impostazione predefinita.

mat

[out] La matrice per la lettura dei dati dal buffer può essere uno qualsiasi dei tre tipi: matrix, matrixf o matrixc.

vec

[out] Il vettore per la lettura dei dati dal buffer può essere di uno qualsiasi dei tre tipi — vector, vectorf or vectorc.

rows=-1

[in] Se il parametro è specificato, è necessario specificare anche il parametro *cols*. Se non vengono specificate le nuove dimensioni della matrice, vengono utilizzate quelle correnti. Se il valore è -1, il numero di righe non cambia.

cols=-1

[in] Se il parametro non è specificato, anche il parametro *rows* deve essere ignorato. La matrice rispetta la regola: o sono specificati entrambi i parametri o nessuno, altrimenti si verificherà un errore. Se vengono specificati entrambi i parametri (*rows* e *cols*), la dimensione della matrice viene modificata. In caso di -1, il numero di colonne non cambia.

size=-1

[in] Se il parametro non è specificato o il suo valore è -1, la lunghezza del vettore non cambia.

Valore restituito

Il numero di elementi di lettura. 0 viene restituito in caso di errore. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

true se una matrice o un vettore vengono gestiti correttamente, altrimenti false.

Nota

Per array uni-dimensionali, il numero dell'elemento, in cui la scrittura di dati in un inizio di buffer OpenCL, viene calcolata tenendo conto delle flags AS_SERIES.

Un array di due o più dimensioni è presentata come uni-dimensionale. In questo caso, *data_offset* è il numero di elementi che devono essere saltati nella presentazione, non il numero di elementi nella prima dimensione.

Esempio calcolo di Pi utilizzando l'equazione:

$$\pi = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16\lim_{N\to\infty}N\sum_{k=0}^{N-1}\frac{1}{4N^2+(2k+1)^2}$$

```
#define  _num_steps         1000000000
#define  _divisor           40000
#define  _step              1.0 / _num_steps
#define  _intrnCnt          _num_steps / _divisor

//+------------------------------------------------------------------+
```

```
//|                                                                  |
//+------------------------------------------------------------------+
string D2S(double arg, int digits) { return DoubleToString(arg, digits); }
string I2S(int arg)                { return IntegerToString(arg); }

//--- Codice programma OpenCL
const string clSource=
  "#define _step "+D2S(_step, 12)+"                    \r\n"
  "#define _intrnCnt "+I2S(_intrnCnt)+"                \r\n"
  "                                                    \r\n"
  "__kernel void Pi( __global double *out )            \r\n"
  "{                                                   \r\n"
  "  int i = get_global_id( 0 );                       \r\n"
  "  double partsum = 0.0;                             \r\n"
  "  double x = 0.0;                                   \r\n"
  "  long from = i * _intrnCnt;                        \r\n"
  "  long to = from + _intrnCnt;                       \r\n"
  "  for( long j = from; j < to; j ++ )                \r\n"
  "  {                                                 \r\n"
  "     x = ( j + 0.5 ) * _step;                       \r\n"
  "     partsum += 4.0 / ( 1. + x * x );               \r\n"
  "  }                                                 \r\n"
  "  out[ i ] = partsum;                               \r\n"
  "}                                                   \r\n";

//+------------------------------------------------------------------+
//| Programma Script funzione start                                  |
//+------------------------------------------------------------------+
int OnStart()
 {
  Print("Pi Calculation: step = "+D2S(_step, 12)+"; _intrnCnt = "+I2S(_intrnCnt));
//--- prepara i contesti OpenCL
  int clCtx;
  if((clCtx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found");
    return(-1);
   }
  int clPrg = CLProgramCreate(clCtx, clSource);
  int clKrn = CLKernelCreate(clPrg, "Pi");
  int clMem=CLBufferCreate(clCtx, _divisor*sizeof(double), CL_MEM_READ_WRITE);
  CLSetKernelArgMem(clKrn, 0, clMem);

  const uint offs[1]  = {0};
  const uint works[1] = {_divisor};
//--- avvia il programma OpenCL
  ulong start=GetMicrosecondCount();
  if(!CLExecute(clKrn, 1, offs, works))
   {
```

```
     Print("CLExecute(clKrn, 1, offs, works) failed! Error ", GetLastError());
     CLFreeAll(clMem, clKrn, clPrg, clCtx);
     return(-1);
    }
//--- ottieni risultati dal dispositivo OpenCL
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
    {
     Print("CLBufferRead(clMem, 0, buffer) failed! Error ", GetLastError());
     CLFreeAll(clMem, clKrn, clPrg, clCtx);
     return(-1);
    }
//--- somma tutti i valori per calcolare Pi
  double Pi=buffer.Sum()*_step;

  double time=(GetMicrosecondCount()-start)/1000.;
  Print("OpenCL: Pi calculated for "+D2S(time, 2)+" ms");
  Print("Pi = "+DoubleToString(Pi, 12));
//--- libera memoria
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- esito positivo
  return(0);
 }
  /*
  Pi Calculation: step = 0.000000001000; _intrnCnt = 25000
  OpenCL: GPU device 'Ellesmere' selected
  OpenCL: Pi calculated for 99.98 ms
  Pi = 3.141592653590
  */
//+------------------------------------------------------------------+
//| Routine ausiliaria per liberare memoria                          |
//+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
```

CLExecute

La funzione esegui un programma OpenCL. Ci sono 3 versioni della funzione:

1. Avvio di funzioni del kernel utilizzando un kernel

```
bool  CLExecute(
   int           kernel                    // Handle al kernel di un programma OpenCL
   );
```

2. Avvio di diverse copie del kernel (funzione OpenCL) con compito descrizione spazio

```
bool  CLExecute(
   int           kernel,                   // Handle al kernel di un programma OpenCL
   uint          work_dim,                 // Dimensione dello spazio dei tasks
   const uint&   global_work_offset[],     // Offset iniziale nello spazio dei tasks
   const uint&   global_work_size[]        // Numero totale dei tasks
   );
```

3. Avvio di diverse copie del kernel (funzione OpenCL) con compito descrizione dello spazio e la specificazione della grandezza del sottoinsieme attività locale del gruppo

```
bool  CLExecute(
   int           kernel,                   // Handle al kernel di un programma OpenCL
   uint          work_dim,                 // Dimensione dello spazio dei tasks
   const uint&   global_work_offset[],     // Offset iniziale nello spazio dei tasks
   const uint&   global_work_size[],       // Numero totale di attività
   const uint&   local_work_size[]         // Numero di attività nel gruppo locale
   );
```

Parametri

kernel

[in] Handle al kernel OpenCL.

work_dim

[in] Dimensione dello spazio tasks.

global_work_offset[]

[in] Offset iniziale nello spazio tasks.

global_work_size[]

[in] Dimensione di un sottoinsieme di attività.

local_work_size[]

[in] La grandezza del sottoinsieme attività locale del gruppo.

Valore restituito

Restituisce vero se ha successo, altrimenti restituisce false. Per ulteriori informazioni sull'errore, utilizzare la funzione GetLastError().

Nota

Considerare l'uso dei parametri nel seguente esempio:

- *work_dim* specifies *work_items[]* dimensione array descrivente l'attività. Se *work_dim*=3, verrà usato un array tri-dimensionale *work_items[N1, N2, N3]*.
- *global_work_size[]* contiene i valori che impostano la grandezza array di *work_items[]*. Se work_dim=3, *global_work_size[3]* l'array può essere {40, 100, 320}. Allora abbiamo *work_items[40, 100, 320]*. Così, il numero totale di attività è 40 x 100 x 320 = 1 280 000.
- *local_work_size[]* imposta il sottoinsieme delle attività che verranno eseguite dal kernel specificato, del programma OpenCL. La sua dimensione è uguale alla dimensione *work_items[]* e consente di dividere il subset del task comune in subsets più piccoli senza la perdita del resto nella divisione. Infatti, le dimensioni dell'array *local_work_size[]* devono essere selezionate in modo da dividere il task globale *work_items[]* in subsets più piccoli. In questo esempio *local_work_size[3] ={10, 10, 10}* sarà OK, come *work_items[40, 100, 320]* può essere raccolto dall' array *local_items[10, 10, 10]* senza il resto della divisione.

CLExecutionStatus

Restituisce lo stato di esecuzione del programma OpenCL.

```
int  CLExecutionStatus(
   int   kernel            // handle al kernel di un programma OpenCL
   );
```

Parametri

kernel

[in] Handle al kernel del programma OpenCL.

Valore di Ritorno

Restituisce lo stato del programma OpenCL. Il valore può essere uno dei seguenti:

- CL_COMPLETE=0 - programma completo,
- CL_RUNNING=1 - sta girando,
- CL_SUBMITTED=2 - inviato per l'esecuzione,
- CL_QUEUED=3 - in coda,
- -1 (meno uno) - errore avvento mentre si eseguiva CLExecutionStatus().

Lavorare con i database

Le funzioni per lavorare con i database usano il popolare e facile-da-usare motore SQLite. Una caratteristica importante di questo motore è che l'intero database è inserito in un singolo file standard situato sul PC dell'utente.

The functions allow for convenient creation of tables, adding data to them, performing modifications and sampling using simple SQL requests:

- receiving trading history and quotes from any formats,
- saving optimization and test results,
- preparing and exchanging data with other analysis packages,
- storing MQL5 application settings and status.

Queries allow using statistical and mathematical functions.

The functions for working with databases allow you to replace the most repetitive large data array handling operations with SQL requests, so that it is often possible to use the DatabaseExecute/DatabasePrepare calls instead of programming complex loops and comparisons. Use the DatabaseReadBind function to conveniently obtain query results in a ready-made structure. The function allows reading all record fields at once within a single call.

To accelerate reading, writing and modification, a database can be opened/created in RAM with the DATABASE_OPEN_MEMORY flag, although such a database is available only to a specific application and is not shared. When working with databases located on the hard disk, bulk data inserts/changes should be wrapped in transactions using DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollback. This accelerates the process hundreds of times.

Per iniziare a lavorare con le funzioni, leggi l'articolo SQLite: gestione nativa dei database SQL in MQL5.

Funzione	Azione
DatabaseOpen	Apre o crea un database in un file specificato
DatabaseClose	Chiude un database
DatabaseImport	Importa i dati da un file in una tabella
DatabaseExport	Esporta una tabella o un risultato di esecuzione di una richiesta SQL in un file CSV
DatabasePrint	Stampa una tabella o un risultato di esecuzione di una richiesta SQL nel journal degli experts
DatabaseTableExists	Verifica la presenza della tabella in un database
DatabaseExecute	Esegue una richiesta ad un database specificato
DatabasePrepare	Crea un handle di una richiesta, che può quindi essere eseguita utilizzando DatabaseRead()

Funzione	Azione
DatabaseReset	Reimposta una richiesta, come dopo aver chiamato DatabasePrepare()
DatabaseBind	Imposta un valore di parametro in una richiesta
DatabaseBindArray	Imposta un array come valore di parametro
DatabaseRead	Passa alla voce successiva a seguito di una richiesta
DatabaseFinalize	Rimuove una richiesta creata in DatabasePrepare()
DatabaseTransactionBegin	Inizia l'esecuzione della transazione
DatabaseTransactionCommit	Completa l'esecuzione della transazione
DatabaseTransactionRollback	Ripristina le transazioni
DatabaseColumnsCount	Ottiene il numero di campi in una richiesta
DatabaseColumnName	Ottiene un nome campo per indice
DatabaseColumnType	Ottiene un tipo di campo per indice
DatabaseColumnSize	Ottiene una dimensione del campo in byte
DatabaseColumnText	Ottiene un valore di campo come stringa dal record corrente
DatabaseColumnInteger	Ottiene il valore del tipo int dal record corrente
DatabaseColumnLong	Ottiene il valore di tipo long dal record corrente
DatabaseColumnDouble	Ottiene il valore di tipo double dal record corrente
DatabaseColumnBlob	Ottiene un valore di campo come un array dal record corrente

Queries allow using statistical and mathematical functions.

Statistical functions:

- mode - mode
- median - median (50th percentile)
- percentile_25 - 25th percentile
- percentile_75
- percentile_90
- percentile_95
- percentile_99
- stddev or stddev_samp — sample standard deviation
- stddev_pop — population standard deviation
- variance or var_samp — sample variance
- var_pop — population variance

Mathematical functions

- acos(X) - arccosine in radians
- acosh(X) - hyperbolic arccosine
- asin(X) - arcsine in radians
- asinh(X) - hyperbolic arcsine
- atan(X) - arctangent in radians
- atan2(X,Y) - arctangent in radians of the X/Y ratio
- atanh(X) - hyperbolic arctangent
- ceil(X) - rounding up to an integer
- ceiling(X) - rounding up to an integer
- cos(X) - angle cosine in radians
- cosh(X) - hyperbolic cosine
- degrees(X) - convert radians into the angle
- exp(X) - exponent
- floor(X) - rounding down to an integer
- ln(X) - natural logarithm
- log(B,X) - logarithm to the indicated base
- log(X) - decimal logarithm
- log10(X) - decimal logarithm
- log2(X) - logarithm to base 2
- mod(X,Y) - remainder of division
- pi() - approximate Pi
- pow(X,Y) - power by the indicated base
- power(X,Y) - power by the indicated base
- radians(X) - convert the angle into radians
- sin(X) - angle sine in radians
- sinh(X) - hyperbolic sine
- sqrt(X) - square root
- tan(X) - angle tangent in radians
- tanh(X) - hyperbolic tangent
- trunc(X) - truncate to an integer closest to 0

Example:

```
select
  count(*) as book_count,
  cast(avg(parent) as integer) as mean,
  cast(median(parent) as integer) as median,
  mode(parent) as mode,
  percentile_90(parent) as p90,
  percentile_95(parent) as p95,
  percentile_99(parent) as p99
from moz_bookmarks;
```

DatabaseOpen

Apre o crea un database in un file specificato.

```
int  DatabaseOpen(
   string  filename,       // nome del file
   uint    flags           // combinazione di flags
   );
```

Parametri

filename

[in] Nome file relativo alla cartella "MQL5\Files".

flags

[In] Combinazione di flags dall'enumerazione ENUM_DATABASE_OPEN_FLAGS.

Valore di ritorno

Se eseguita correttamente, la funzione restituisce l'handle del database, che viene quindi utilizzato per accedere al database. Altrimenti, restituisce INVALID_HANDLE. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INTERNAL_ERROR (4001) - critical runtime error;
- ERR_WRONG_INTERNAL_PARAMETER (4002) - internal error, while accessing the "MQL5\Files" folder;
- ERR_INVALID_PARAMETER (4003) - path to the database file contains an empty string, or an incompatible combination of flags is set;
- ERR_NOT_ENOUGH_MEMORY (4004) - insufficient memory;
- ERR_WRONG_FILENAME (5002) - wrong database file name;
- ERR_TOO_LONG_FILENAME (5003) - absolute path to the database file exceeds the maximum length;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - exceeded the maximum acceptable number of Database objects;
- ERR_DATABASE_CONNECT (5123) - database connection error;
- ERR_DATABASE_MISUSE (5621) - uso non corretto della libreria SQLite.

Nota

Se il parametro *filename* presenta NULL o la stringa vuota "", viene creato un file temporaneo sul disco. Viene automaticamente eliminato dopo aver chiuso la connessione al database.

Se il parametro *filename* presenta ":memory:", il database viene creato nella memoria e viene automaticamente eliminato dopo la chiusura della connessione.

Se il parametro *flags* non presenta nessuno dei flag DATABASE_OPEN_READONLY o DATABASE_OPEN_READWRITE, viene utilizzato il flag DATABASE_OPEN_READWRITE.

Se l'estensione del file non è specificata, viene utilizzata ".sqlite" .

ENUM_DATABASE_OPEN_FLAGS

ID	Descrizione
DATABASE_OPEN_READONLY	Sola lettura
DATABASE_OPEN_READWRITE	Apre per leggere e scrivere
DATABASE_OPEN_CREATE	Crea il file su un disco, se necessario
DATABASE_OPEN_MEMORY	Crea un database nella RAM
DATABASE_OPEN_COMMON	Il file si trova nella cartella comune a tutti i terminali

Guarda anche

DatabaseClose

DatabaseClose

Chiude un database.

```
void  DatabaseClose(
   int  database      // handle database ricevuto in DatabaseOpen
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

Valore di ritorno

Nessuno.

Nota

Dopo aver chiamato DatabaseClose, tutti gli handles di richieste al database vengono automaticamente rimossi e diventano non validi.

Se l'handle non è valido, la funzione imposta l'errore ERR_DATABASE_INVALID_HANDLE. Puoi controllare l'errore usando GetLastError().

Guarda anche

DatabaseOpen, DatabasePrepare

DatabaseImport

Importa i dati da un file in una tabella.

```
long  DatabaseImport(
   int           database,          // handle di database ricevuto in DatabaseOpen
   const string  table,             // nome di una tabella per inserire dati
   const string  filename,          // nome di un file per importare dati
   uint          flags,             // combinazione di flags
   const string  separator,         // separatore di dati
   ulong         skip_rows,         // quante stringhe iniziali saltare
   const string  skip_comments      // stringa di caratteri che definiscono i commenti
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

table

[in] Nome di una tabella in cui aggiungere i dati di un file.

filename

[in] file CSV o archivio ZIP per la lettura dei dati. Il nome può contenere sottodirectory ed è impostato rispetto alla cartella MQL5\Files.

flags

[in] Combinazione di flag dall'enumerazione ENUM_DATABASE_IMPORT_FLAGS.

separator

[in] Separatore di dati nel file CSV.

skip_rows

[in] Numero di stringhe iniziali da saltare durante la lettura dei dati dal file.

skip_comments

[in] Stringa di caratteri per designare stringhe come commenti. Se qualche carattere di *skip_comments* viene rilevato all'inizio di una stringa, tale stringa viene considerata un commento e non viene importata.

Valore di Ritorno

Restituisce il numero di stringhe importate o -1 in caso di errore. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INVALID_PARAMETER (4003) - nessun nome tabella specificato (stringa vuota o NULL);
- ERR_DATABASE_INTERNAL (5120) - errore interno al database;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle di database non valido.

Nota

Se non esiste una tabella denominata *table*, viene generata automaticamente. Nomi e tipi di campi nella tabella creata vengono definiti automaticamente in base ai dati del file.

ENUM_DATABASE_IMPORT_FLAGS

ID	Descrizione
DATABASE_IMPORT_HEADER	La prima riga contiene i nomi dei campi della tabella
DATABASE_IMPORT_CRLF	CRLF (l'impostazione predefinita è LF) è considerata un'interruzione di stringa
DATABASE_IMPORT_APPEND	Aggiunge dati alla fine di una tabella esistente
DATABASE_IMPORT_QUOTED_STRINGS	Valori stringa racchiusi tra virgolette
DATABASE_IMPORT_COMMON_FOLDER	Il file è archiviato nella cartella common di tutti i terminali client \Terminal\Common\File.

Esempio di lettura della tabella dal file creato dal codice d'esempio DatabaseExport:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- ottiene i nomi dei file di testo da scaricare dalla cartella common dei terminal
  string filenames[];
  if(FileSelectDialog("Select a CSV file to download a table", NULL,
                      "Text files (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER, filenames, "data.csv")>0)
   {
    //--- visualizza il nome di ogni file selezionato
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Unknown error while selecting file. Error code ", GetLastError());
      return;
     }
   }
  else
   {
    Print("CSV file not selected");
    return;
   }
//--- crea o apre un database
  string db_filename="test.sqlite";
```

```
  int db=DatabaseOpen(db_filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
//--- controlla se la tabella TEST esiste
  if(DatabaseTableExists(db, "TEST"))
   {
    //--- rimuove la tabella TEST
    if(!DatabaseExecute(db, "DROP TABLE IF EXISTS TEST"))
     {
      Print("Failed to drop the TEST table with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   }
  //--- importa le voci dal file alla tabella TEST
  long imported=DatabaseImport(db, "TEST", csv_filename, DATABASE_IMPORT_HEADER|DATABA
  if(imported>0)
   {
    Print(imported," lines imported in table TEST");
    DatabasePrint(db,"SELECT * FROM TEST",DATABASE_PRINT_NO_INDEX);
   }
  else
    {
     Print("DatabaseImport() failed. Error ",GetLastError());
    }
//--- chiude il file del database e lo comunica
  DatabaseClose(db);
  PrintFormat("Database: %s closed", db_filename);
 }
```

Guarda anche

DatabaseOpen, DatabasePrint

DatabaseExport

Esporta una tabella o un risultato di esecuzione di una richiesta SQL in un file CSV. Il file viene creato in codifica UTF-8.

```
long  DatabaseExport(
   int           database,          // handle di database ricevuto in DatabaseOpen
   const string  table_or_sql,      // un nome tabella o una richiesta SQL
   const string  filename,          // un nome di un file CSV per l'esportazione dei
   uint          flags,             // combinazione di bandiere
   const string  separator          // separatore di dati nel file CSV
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

table_or_sql

[in] Il nome di una tabella o un testo di una richiesta SQL i cui risultati devono essere esportati in un file specificato.

filename

[in] Il nome file per l'esportazione dei dati. Il percorso è impostato rispetto alla cartella MQL5\Files.

flags

[in] Combinazione di flag dall'enumerazione ENUM_DATABASE_EXPORT_FLAGS.

separator

[in] Separatore di dati. Se si specifica NULL, viene utilizzato come separatore il carattere di tabulazione '\t'. Una stringa vuota "" è considerata un separatore valido ma il file CSV ottenuto non può essere letto come una tabella - è considerato come un insieme di stringhe.

Valore di Ritorno

Restituisce il numero di voci esportate o un valore negativo in caso di errore. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INTERNAL_ERROR (4001) - errore critico di runtime;
- ERR_INVALID_PARAMETER (4003) - il percorso del file di database contiene una stringa vuota o viene impostata una combinazione incompatibile di flag;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insufficiente;
- ERR_FUNCTION_NOT_ALLOWED(4014) - la pipe specificata non è consentita;
- ERR_PROGRAM_STOPPED(4022) - operazione annullata (programma MQL interrotto);
- ERR_WRONG_FILENAME (5002) - nome file non valido;
- ERR_TOO_LONG_FILENAME (5003) - il percorso assoluto del file supera la lunghezza massima;
- ERR_CANNOT_OPEN_FILE(5004) - impossibile aprire il file per la scrittura;
- ERR_FILE_WRITEERROR(5026) - impossibile scrivere nel file;
- ERR_DATABASE_INTERNAL (5120) - errore interno al database;

- ERR_DATABASE_INVALID_HANDLE (5121) - handle di database non valido;
- ERR_DATABASE_QUERY_PREPARE(5125) - errore di generazione richiesta;
- ERR_DATABASE_QUERY_NOT_READONLY - la richiesta di sola lettura è consentita.

Nota

Se i risultati della richiesta vengono esportati, la richiesta SQL dovrebbe iniziare con "SELECT" o "select". In altre parole, la richiesta SQL non può modificare lo stato del database, altrimenti DatabaseExport() fallisce con un errore.

I valori delle stringhe del database possono contenere il carattere di conversione ('\r' o '\r \n'), nonché il set di caratteri del separatore di valori nel parametro *separator*. In questo caso, assicurarsi di utilizzare il flag DATABASE_EXPORT_QUOTED_STRINGS nel parametro 'flags'. Se questo flag è presente, tutte le stringhe visualizzate sono racchiuse tra virgolette doppie. Se una stringa contiene una virgoletta doppia, viene sostituita da due virgolette doppie.

ENUM_DATABASE_EXPORT_FLAGS

ID	Descrizione
DATABASE_EXPORT_HEADER	Visualizza i nomi dei campi nella prima stringa
DATABASE_EXPORT_INDEX	Visualizza gli indici delle stringhe
DATABASE_EXPORT_NO_BOM	Non inserire il contrassegno BOM all'inizio del file (di default BOM è inserito)
DATABASE_EXPORT_CRLF	Usa CRLF per l'interruzione di stringa (l'impostazione predefinita è LF)
DATABASE_EXPORT_APPEND	Aggiunge dati alla fine di un file esistente (per impostazione predefinita, il file viene sovrascritto). Se il file non esiste, verrà creato.
DATABASE_EXPORT_QUOTED_STRINGS	Visualizza i valori delle stringhe tra doppie virgolette.
DATABASE_EXPORT_COMMON_FOLDER	Un file CSV viene creato nella cartella common di tutti i terminali client \Terminal\Common\File.

Esempio:

```
input int InpRates=100;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  MqlRates rates[];
```

```
//--- ricordare l'ora di inizio prima della ricezione delle barre
  ulong start=GetMicrosecondCount();
//--- richiedere le ultime 100 barre su H1
  if(CopyRates(Symbol(), PERIOD_H1, 1, InpRates, rates)<InpRates)
    {
     Print("CopyRates() failed,, Error ", GetLastError());
     return;
    }
  else
    {
     //---quante barre sono state ricevute e quanto tempo ci è voluto per riceverle
     PrintFormat("%s: CopyRates received %d bars in %d ms ",
                 _Symbol, ArraySize(rates), (GetMicrosecondCount()-start)/1000);
    }
//--- impostare il nome del file per la memorizzazione del database
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)+"_"+TimeToString(TimeCurrent())+
  StringReplace(filename, ":", "-"); // ":" character is not allowed in file names
//--- aprire/creare il database nella cartella common del terminale
  int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE|DATABASE_
  if(db==INVALID_HANDLE)
    {
     Print("Database: ", filename, " open failed with code ", GetLastError());
     return;
    }
  else
     Print("Database: ", filename, " opened successfully");

//--- controlla se la tabella RATES esiste
  if(DatabaseTableExists(db, "RATES"))
    {
     //--- rimuove la tabella RATES
     if(!DatabaseExecute(db, "DROP TABLE IF EXISTS RATES"))
       {
        Print("Failed to drop the RATES table with code ", GetLastError());
        DatabaseClose(db);
        return;
       }
    }
//--- crea la tabella RATES
  if(!DatabaseExecute(db, "CREATE TABLE RATES("
                          "SYMBOL             CHAR(10),"
                          "TIME               INT NOT NULL,"
                          "OPEN               REAL,"
                          "HIGH               REAL,"
                          "LOW                REAL,"
                          "CLOSE              REAL,"
                          "TICK_VOLUME        INT,"
                          "SPREAD             INT,"
                          "REAL_VOLUME        INT);"))
```

```
    {
     Print("DB: ", filename, " create table RATES with code ", GetLastError());
     DatabaseClose(db);
     return;
    }
//--- visualizza l'elenco di tutti i campi della tabella RATES
  if(DatabasePrint(db, "PRAGMA TABLE_INFO(RATES)", 0)<0)
    {
     PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(RATES)\") failed, error code=%d at
     DatabaseClose(db);
     return;
    }
//--- creare una richiesta parametrizzata per aggiungere barre alla tabella RATES
  string sql="INSERT INTO RATES (SYMBOL,TIME,OPEN,HIGH,LOW,CLOSE,TICK_VOLUME,SPREAD,RE
             " VALUES (?1,?2,?3,?4,?5,?6,?7,?8,?9)"; // parametri richiesti
  int request=DatabasePrepare(db, sql);
  if(request==INVALID_HANDLE)
    {
     PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
     Print("SQL request: ", sql);
     DatabaseClose(db);
     return;
    }
//--- impostare il valore del primo parametro di richiesta
  DatabaseBind(request, 0, _Symbol);
//--- ricordare l'ora di inizio prima dell'aggiunta delle barre
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0; i<total; i++)
    {
     //--- impostare i valori dei restanti parametri prima di aggiungere la voce
     ResetLastError();
     if(!DatabaseBind(request, 1, rates[i].time))
       {
        PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
        PrintFormat("Bar #%d line=%d", i+1, __LINE__);
        request_error=true;
        break;
       }
     //--- se la precedente chiamata a DatabaseBind() ha avuto esito positivo, impostar
     if(!request_error && !DatabaseBind(request, 2, rates[i].open))
       {
        PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
        PrintFormat("Bar #%d line=%d", i+1, __LINE__);
        request_error=true;
        break;
       }
```

```
   if(!request_error && !DatabaseBind(request, 3, rates[i].high))
    {
     PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
     PrintFormat("Bar #%d line=%d", i+1, __LINE__);
     request_error=true;
     break;
    }
   if(!request_error && !DatabaseBind(request, 4, rates[i].low))
    {
     PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
     PrintFormat("Bar #%d line=%d", i+1, __LINE__);
     request_error=true;
     break;
    }
   if(!request_error && !DatabaseBind(request, 5, rates[i].close))
    {
     PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
     PrintFormat("Bar #%d line=%d", i+1, __LINE__);
     request_error=true;
     break;
    }
   if(!request_error && !DatabaseBind(request, 6, rates[i].tick_volume))
    {
     PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
     PrintFormat("Bar #%d line=%d", i+1, __LINE__);
     request_error=true;
     break;
    }
   if(!request_error && !DatabaseBind(request, 7, rates[i].spread))
    {
     PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
     PrintFormat("Bar #%d line=%d", i+1, __LINE__);
     request_error=true;
     break;
    }
   if(!request_error && !DatabaseBind(request, 8, rates[i].real_volume))
    {
     PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
     PrintFormat("Bar #%d line=%d", i+1, __LINE__);
     request_error=true;
     break;
    }

   //--- eseguire una richiesta di inserimento della voce e verificare la presenza di
   if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_MO
    {
     PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
     DatabaseFinalize(request);
     request_error=true;
```

```
      break;
     }
    //--- resettare la richiesta prima del successivo aggiornamento del parametro
    if(!request_error && !DatabaseReset(request))
     {
      PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
      DatabaseFinalize(request);
      request_error=true;
      break;
     }
   } //--- finito tutte le barre attraversate

//--- stato delle transazioni
  if(request_error)
   {
    PrintFormat("Table RATES: failed to add %d bars ", ArraySize(rates));
    DatabaseTransactionRollback(db);
    DatabaseClose(db);
    return;
   }
  else
   {
    DatabaseTransactionCommit(db);
    PrintFormat("Table RATES: added %d bars in %d ms",
                ArraySize(rates), (GetMicrosecondCount()-start)/1000);
   }
//--- salva la tabella RATES in un file CSV
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db, "SELECT * FROM RATES", csv_filename, DATABASE_EXPORT_F
  if(saved>0)
    Print("Table RATES saved in ", Symbol(), ".csv");
  else
    Print("DatabaseExport() failed. Error ", GetLastError());
//--- chiude il file del database e lo comunica
  DatabaseClose(db);
  PrintFormat("Database: %s created and closed", filename);
```

Guarda anche

DatabasePrint, DatabaseImport

DatabasePrint

Stampa una tabella o un risultato di esecuzione di una richiesta SQL nel journal degli experts.

```
long  DatabasePrint(
   int           database,          // handle di database ricevuto in DatabaseOpen
   const string  table_or_sql,      // una tabella o una richiesta SQL
   uint          flags              // combinazione di flags
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

table_or_sql

[in] Il nome di una tabella o un testo di una richiesta SQL i cui risultati sono visualizzati nel journal degli experts.

flags

[in] Combinazione di flag che definiscono la formattazione dell'output. I flag sono definiti come segue:

DATABASE_PRINT_NO_HEADER - non mostra i nomi delle colonne della tabella (nomi dei campi)
DATABASE_PRINT_NO_INDEX - non mostra gli indici di stringa
DATABASE_PRINT_NO_FRAME - non mostra il frame che separa intestazione e dati
DATABASE_PRINT_STRINGS_RIGHT - allinea le stringhe a destra.

Se flags=0, vengono visualizzate le colonne e le stringhe, l'intestazione e i dati sono separati dal frame, mentre le stringhe sono allineate a sinistra.

Valore di Ritorno

Restituisce il numero di stringhe esportate o -1 in caso di errore. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INTERNAL_ERROR (4001) - errore critico di runtime;
- ERR_NOT_ENOUGH_MEMORY (4004) - memoria insufficiente;
- ERR_DATABASE_INTERNAL (5120) - errore interno al database;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle di database non valido;

Nota

Se il journal visualizza i risultati della richiesta, la richiesta SQL dovrebbe iniziare con "SELECT" o "select". In altre parole, la richiesta SQL non può modificare lo stato del database, altrimenti DatabasePrint() fallisce con un errore.

Esempio:

```
//+------------------------------------------------------------------+
//| Funzione di start del programma script                           |
//+------------------------------------------------------------------+
void OnStart()
  {

```

```
   string filename="departments.sqlite";
//--- crea o apre il database nella cartella terminale comune
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }

//--- crea la tabella COMPANY
   if(!CreataTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- crea la tabella DEPARTMENT
   if(!CreataTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- mostra l'elenco di tutti i campi nelle tabelle COMPANY e DEPARTMENT
   PrintFormat("Prova a stampare la richiesta \"PRAGMA TABLE_INFO(COMPANY);PRAGMA TABL
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)", 0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO()\") fallito, codice errore=%d",
      DatabaseClose(db);
      return;
     }
//--- mostra la tabella COMPANY nel registro
   PrintFormat("Prova a stampare la richiesta \"SELECT * da COMPANY\"");
   if(DatabasePrint(db, "SELECT * from COMPANY", 0)<0)
     {
      Print("DatabasePrint fallito con codice", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- richiesta di testo per combinare le tabelle COMPANY e DEPARTMENT
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT "
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- mostra la tabella che combina il risultato
   PrintFormat("Prova a stampare la richiesta\"SELECT EMP_ID, NAME, DEPT FROM COMPANY
   if(DatabasePrint(db, request, 0)<0)
     {
      Print("DatabasePrint fallito con codice", GetLastError());
      DatabaseClose(db);
      return;
     }
```

```
//--- chiude il database
   DatabaseClose(db);
  }
/*
Conclusione:
Try to print request "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)"
#| cid name    type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID      INT            1             1
2|   1 NAME    TEXT           1             0
3|   2 AGE     INT            1             0
4|   3 ADDRESS CHAR(50)       0             0
5|   4 SALARY  REAL           0             0
#| cid name   type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID     INT            1             1
2|   1 DEPT   CHAR(50)       1             0
3|   2 EMP_ID INT            1             0
Try to print request "SELECT * from COMPANY"
#| ID NAME  AGE ADDRESS     SALARY
-+-------------------------------
1|  1 Paul   32 California 25000.0
2|  2 Allen  25 Texas      15000.0
3|  3 Teddy  23 Norway     20000.0
4|  4 Mark   25 Rich-Mond  65000.0
5|  5 David  27 Texas      85000.0
6|  6 Kim    22 South-Hall 45000.0
7|  7 James  24 Houston    10000.0
Try to print request "SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMEN
#| EMP_ID NAME  DEPT
-+-------------------------
1|      1 Paul  IT Billing
2|      2 Allen Engineering
3|        Teddy
4|        Mark
5|        David
6|        Kim
7|      7 James Finance
*/
//+------------------------------------------------------------------+
//| Crea la tabella COMPANY                                          |
//+------------------------------------------------------------------+
bool CreateTableCompany(int database)
  {
//--- se esiste la tabella COMPANY, eliminala
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- elimina la tabella
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
```

```
         {
          Print("Fallimento nell'eliminare la tabella COMPANY con codice", GetLastError
          return(false);
         }
      }
//--- crea la tabella COMPANY
   if(!DatabaseExecute(database, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
      {
       Print("DB: creazione tabella COMPANY fallita con codice ", GetLastError());
       return(false);
      }

//--- inserisci i dati nella tabella COMPANY
   if(!DatabaseExecute(database, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VAI
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'I
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'P
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'J
      {
       Print("inserimento COMPANY fallito con codice", GetLastError());
       return(false);
      }
//--- successo
   return(true);
  }
//+------------------------------------------------------------------+
//| Creare la tabella DEPARTMENT                                     |
//+------------------------------------------------------------------+
bool CreateTableDepartment(int database)
  {
//--- se esiste la tabella DEPARTMENT, eliminala
   if(DatabaseTableExists(database, "DEPARTMENT"))
      {
       //--- elimina la tabella
       if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
         {
          Print("Fallimento nell' eliminare la tabella DEPARTMENT con codice", GetLastE
          return(false);
         }
      }
//--- crea la tabella DEPARTMENT
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
```

```
                        "ID      INT PRIMARY KEY   NOT NULL,"
                        "DEPT    CHAR(50)          NOT NULL,"
                        "EMP_ID  INT               NOT NULL);"))
     {
      Print("DB: creazione tabella DEPARTMENT fallita con codice ", GetLastError());
      return(false);
     }

//--- inserisci i dati nella tabella DEPARTMENT
   if(!DatabaseExecute(database, "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (1, 
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (2, 'Engineerir
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (3, 'Finance', 
     {
      Print("inserimento DEPARTMENT fallito con codice ", GetLastError());
      return(false);
     }
//--- successo
   return(true);
  }
//+------------------------------------------------------------------+
```

Guarda anche

DatabaseExport, DatabaseImport

DatabaseTableExists

Verifica la presenza della tabella nel database.

```
bool  DatabaseTableExists(
   int     database,      // handle di database ricevuto in DatabaseOpen
   string  table          // nome tabella
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

table

[in] Nome tabella.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INVALID_PARAMETER (4003) - no table name specified (empty string or NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INTERNAL (5120) - internal database error;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_EXECUTE (5124) - request execution error;
- ERR_DATABASE_NO_MORE_DATA (5126) - no table exists (not an error, normal completion).

Guarda anche

DatabasePrepare, DatabaseFinalize

DatabaseExecute

Esegue una richiesta ad un database specificato.

```
bool  DatabaseExecute(
   int     database,      // handle di database ricevuto in DatabaseOpen
   string  sql            // Richiesta SQL
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

sql

[in] Richiesta SQL.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INTERNAL_ERROR (4001) - critical runtime error;
- ERR_INVALID_PARAMETER (4003) - sql parameter contains an empty string;
- ERR_NOT_ENOUGH_MEMORY (4004) - insufficient memory;
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INTERNAL (5120) - internal database error;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_EXECUTE (5124) - request execution error.

Example:

```
//--- symbol statistics
struct Symbol_Stats
  {
   string            name;             // symbol name
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and co
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swa
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
```

```
  };

//--- Magic Number statistics
struct Magic_Stats
  {
   long              magic;            // EA's Magic Number
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and cor
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

//--- entry hour statistics
struct Hour_Stats
  {
   char              hour_in;          // market entry hour
   int               trades;           // number of trades in this entry hour
   double            volume;           // volume of trades in this entry hour
   double            gross_profit;     // total profit in this entry hour
   double            gross_loss;       // total loss in this entry hour
   double            net_profit;       // net profit taking into account swaps and cor
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

int ExtDealsTotal=0;;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
```

```
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_stats.sqlite";
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
   PrintFormat("Deals in the trading history: %d ", ExtDealsTotal);

//--- get trading statistics per symbols
   int request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gro
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-
                               "FROM "
                               "   ("
                               "   SELECT SYMBOL,"
                               "   sum(case when entry =1 then 1 else 0 end) as trades
                               "   sum(case when profit > 0 then profit else 0 end) as
                               "   sum(case when profit < 0 then profit else 0 end) as
                               "   sum(swap) as total_swap,"
                               "   sum(commission) as total_commission,"
                               "   sum(profit) as total_profit,"
                               "   sum(profit+swap+commission) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_
                               "   sum(case when profit < 0 then 1 else 0 end) as loss
                               "   FROM DEALS "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY SYMBOL"
                               "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Symbol_Stats stats[], symbol_stats;
   ArrayResize(stats, ExtDealsTotal);
   int i=0;
```

```
//--- get records from request results
   for(; DatabaseReadBind(request, symbol_stats) ; i++)
     {
      stats[i].name=symbol_stats.name;
      stats[i].trades=symbol_stats.trades;
      stats[i].gross_profit=symbol_stats.gross_profit;
      stats[i].gross_loss=symbol_stats.gross_loss;
      stats[i].total_commission=symbol_stats.total_commission;
      stats[i].total_swap=symbol_stats.total_swap;
      stats[i].total_profit=symbol_stats.total_profit;
      stats[i].net_profit=symbol_stats.net_profit;
      stats[i].win_trades=symbol_stats.win_trades;
      stats[i].loss_trades=symbol_stats.loss_trades;
      stats[i].expected_payoff=symbol_stats.expected_payoff;
      stats[i].win_percent=symbol_stats.win_percent;
      stats[i].loss_percent=symbol_stats.loss_percent;
      stats[i].average_profit=symbol_stats.average_profit;
      stats[i].average_loss=symbol_stats.average_loss;
      stats[i].profit_factor=symbol_stats.profit_factor;
     }
   ArrayResize(stats, i);
   Print("Trade statistics by Symbol");
   ArrayPrint(stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- get trading statistics for Expert Advisors by Magic Numbers
   request=DatabasePrepare(db, "SELECT r.*,"
                           "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                           "   (case when r.trades != 0 then r.win_trades*100.0/r.trad
                           "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                           "   r.gross_profit/r.win_trades as average_profit,"
                           "   r.gross_loss/r.loss_trades as average_loss,"
                           "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                           "FROM "
                           "   ("
                           "   SELECT MAGIC,"
                           "   sum(case when entry =1 then 1 else 0 end) as trades,"
                           "   sum(case when profit > 0 then profit else 0 end) as gro
                           "   sum(case when profit < 0 then profit else 0 end) as gro
                           "   sum(swap) as total_swap,"
                           "   sum(commission) as total_commission,"
                           "   sum(profit) as total_profit,"
                           "   sum(profit+swap+commission) as net_profit,"
                           "   sum(case when profit > 0 then 1 else 0 end) as win_trad
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM DEALS "
                           "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
```

```
                              "   GROUP BY MAGIC"
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Magic_Stats EA_stats[], magic_stats;
   ArrayResize(EA_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, magic_stats) ; i++)
     {
      EA_stats[i].magic=magic_stats.magic;
      EA_stats[i].trades=magic_stats.trades;
      EA_stats[i].gross_profit=magic_stats.gross_profit;
      EA_stats[i].gross_loss=magic_stats.gross_loss;
      EA_stats[i].total_commission=magic_stats.total_commission;
      EA_stats[i].total_swap=magic_stats.total_swap;
      EA_stats[i].total_profit=magic_stats.total_profit;
      EA_stats[i].net_profit=magic_stats.net_profit;
      EA_stats[i].win_trades=magic_stats.win_trades;
      EA_stats[i].loss_trades=magic_stats.loss_trades;
      EA_stats[i].expected_payoff=magic_stats.expected_payoff;
      EA_stats[i].win_percent=magic_stats.win_percent;
      EA_stats[i].loss_percent=magic_stats.loss_percent;
      EA_stats[i].average_profit=magic_stats.average_profit;
      EA_stats[i].average_loss=magic_stats.average_loss;
      EA_stats[i].profit_factor=magic_stats.profit_factor;
     }
   ArrayResize(EA_stats, i);
   Print("Trade statistics by Magic Number");
   ArrayPrint(EA_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accou
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- deals cannot be transformed to trades using a simple method through transa
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
```

```
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,HOUR_IN,TICKET,TYPE,VOLUME,S
                              "SELECT "
                              "   d1.time as time_in,"
                              "   d1.hour as hour_in,"
                              "   d1.position_id as ticket,"
                              "   d1.type as type,"
                              "   d1.volume as volume,"
                              "   d1.symbol as symbol,"
                              "   d1.price as price_in,"
                              "   d2.time as time_out,"
                              "   d2.price as price_out,"
                              "   d1.commission+d2.commission as commission,"
                              "   d2.swap as swap,"
                              "   d2.profit as profit "
                              "FROM DEALS d1 "
                              "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                              "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the table TRADES failed with code ", GetLastError());
         return;
        }

//--- get trading statistics by market entry hours
   request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.trad
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                               "FROM "
                               "   ("
                               "   SELECT HOUR_IN,"
                               "   count() as trades,"
                               "   sum(volume) as volume,"
                               "   sum(case when profit > 0 then profit else 0 end) as gro
                               "   sum(case when profit < 0 then profit else 0 end) as gro
                               "   sum(profit) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_trad
                               "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                               "   FROM TRADES "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY HOUR_IN"
```

```
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Hour_Stats hours_stats[], h_stats;
   ArrayResize(hours_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, h_stats) ; i++)
     {
      hours_stats[i].hour_in=h_stats.hour_in;
      hours_stats[i].trades=h_stats.trades;
      hours_stats[i].volume=h_stats.volume;
      hours_stats[i].gross_profit=h_stats.gross_profit;
      hours_stats[i].gross_loss=h_stats.gross_loss;
      hours_stats[i].net_profit=h_stats.net_profit;
      hours_stats[i].win_trades=h_stats.win_trades;
      hours_stats[i].loss_trades=h_stats.loss_trades;
      hours_stats[i].expected_payoff=h_stats.expected_payoff;
      hours_stats[i].win_percent=h_stats.win_percent;
      hours_stats[i].loss_percent=h_stats.loss_percent;
      hours_stats[i].average_profit=h_stats.average_profit;
      hours_stats[i].average_loss=h_stats.average_loss;
      hours_stats[i].profit_factor=h_stats.profit_factor;
     }
   ArrayResize(hours_stats, i);
   Print("Trade statistics by entry hour");
   ArrayPrint(hours_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
   return;
  }
/*
Deals in the trading history: 2771
Trade statistics by Symbol
      [name] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tot
[0] "AUDUSD"      112      503.20000   -568.00000           -8.83000    -24.64000
[1] "EURCHF"      125      607.71000   -956.85000          -11.77000    -45.02000
[2] "EURJPY"      127     1078.49000  -1057.83000          -10.61000    -45.76000
[3] "EURUSD"      233     1685.60000  -1386.80000          -41.00000    -83.76000
[4] "GBPCHF"      125     1881.37000  -1424.72000          -22.60000    -51.56000
[5] "GBPJPY"      127     1943.43000  -1776.67000          -18.84000    -52.46000
```

```
[6] "GBPUSD"      121     1668.50000  -1438.20000           -7.96000    -49.93000
[7] "USDCAD"       99      405.28000   -475.47000           -8.68000    -31.68000
[8] "USDCHF"      206     1588.32000  -1241.83000          -17.98000    -65.92000
[9] "USDJPY"      107      464.73000   -730.64000          -35.12000    -34.24000

Trade statistics by Magic Number
    [magic] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tota
[0]     100      242     2584.80000  -2110.00000          -33.36000    -93.53000
[1]     200      254     3021.92000  -2834.50000          -29.45000    -98.22000
[2]     300      250     2489.08000  -2381.57000          -34.37000    -96.58000
[3]     400      224     1272.50000  -1283.00000          -24.43000    -64.80000
[4]     500      198     1141.23000  -1051.91000          -27.66000    -63.36000
[5]     600      214     1317.10000  -1396.03000          -34.12000    -68.48000

Trade statistics by entry hour
     [hour_in] [trades] [volume] [gross_profit] [gross_loss] [net_profit] [win_trades]
[ 0]         0       50  5.00000      336.51000   -747.47000   -410.96000           21
[ 1]         1       20  2.00000      102.56000    -57.20000     45.36000           12
[ 2]         2        6  0.60000       38.55000    -14.60000     23.95000            5
[ 3]         3       38  3.80000      173.84000   -200.15000    -26.31000           22
[ 4]         4       60  6.00000      361.44000   -389.40000    -27.96000           27
[ 5]         5       32  3.20000      157.43000   -179.89000    -22.46000           20
[ 6]         6       18  1.80000       95.59000   -162.33000    -66.74000           11
[ 7]         7       14  1.40000       38.48000   -134.30000    -95.82000            9
[ 8]         8       42  4.20000      368.48000   -322.30000     46.18000           24
[ 9]         9      118 11.80000     1121.62000   -875.21000    246.41000           72
[10]        10      206 20.60000     2280.59000  -2021.80000    258.79000          115
[11]        11      138 13.80000     1377.02000   -994.18000    382.84000           84
[12]        12      152 15.20000     1247.56000  -1463.80000   -216.24000           84
[13]        13       64  6.40000      778.27000   -516.22000    262.05000           36
[14]        14       62  6.20000      536.93000   -427.47000    109.46000           38
[15]        15       50  5.00000      699.92000   -413.00000    286.92000           28
[16]        16       88  8.80000      778.55000   -514.00000    264.55000           51
[17]        17       76  7.60000      533.92000  -1019.46000   -485.54000           44
[18]        18       52  5.20000      237.17000   -246.78000     -9.61000           24
[19]        19       52  5.20000      407.67000   -150.36000    257.31000           30
[20]        20       18  1.80000       65.92000    -89.09000    -23.17000            9
[21]        21       10  1.00000       41.86000    -32.38000      9.48000            7
[22]        22       14  1.40000       45.55000    -83.72000    -38.17000            6
[23]        23        2  0.20000        1.20000     -1.90000     -0.70000            1
*/

//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
```

```
      {
       return(false);
      }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "HOUR        INT,"
                          "REASON      INT);"))
        {
         Print("DB: create the DEALS table  failed with code ", GetLastError());
         return(false);
        }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   ExtDealsTotal=HistoryDealsTotal();
//--- add deals to the table
   if(!InsertDeals(database))
      return(false);
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;          // deal ticket
   long     order_ticket;         // the ticket of the order by which the deal was exec
   long     position_ticket;      // ID of the position to which the deal belongs
   datetime time;                 // deal execution time
   long     type ;                // deal type
   long     entry ;               // deal direction
   string   symbol;               // the symbol fro which the deal was executed
   double   volume;               // operation volume
   double   price;                // price
   double   profit;               // financial result
   double   swap;                 // swap
   double   commission;           // commission
   long     magic;                // Magic number (Expert Advisor ID)
   long     reason;               // deal execution reason or source
   char     hour;                 // deal execution hour
   MqlDateTime time_strusture;
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      TimeToStruct(time, time_strusture);
      hour= (char)time_strusture.hour;
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
```

```
                                        "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                        deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastErro
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "HOUR_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
```

```
         }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Guarda anche

DatabasePrepare, DatabaseFinalize

DatabasePrepare

Crea un handle di una richiesta, che può quindi essere eseguita utilizzando DatabaseRead().

```
int  DatabasePrepare(
   int     database,      // handle di database ricevuto in DatabaseOpen
   string  sql,           // richiesta SQL
           ...            // parametri della richiesta
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

sql

[in] Richiesta SQL che può contenere parametri sostituiti automaticamente denominati ?1, ?2,...

...

[in] Parametri di richiesta sostituiti automaticamente.

Valore di ritorno

In caso di successo, la funzione restituisce un handle per la richiesta SQL. Altrimenti, restituisce INVALID_HANDLE. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INVALID_PARAMETER (4003) - path to the database file contains an empty string, or an incompatible combination of flags is set;
- ERR_NOT_ENOUGH_MEMORY (4004) - insufficient memory;
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - exceeded the maximum acceptable number of Database objects;
- ERR_DATABASE_PREPARE (5125) - request generation error.

Nota

La funzione DatabasePrepare() non esegue una richiesta ad un database. Lo scopo è verificare i parametri della richiesta e restituire l'handle per l'esecuzione della richiesta SQL in base ai risultati della verifica. La richiesta stessa viene impostata durante la prima chiamata di DatabaseRead().

Example:

```
//--- Structure to store the deal
struct Deal
  {
   ulong             ticket;           // DEAL_TICKET
   long              order_ticket;     // DEAL_ORDER
   long              position_ticket;  // DEAL_POSITION_ID
   datetime          time;             // DEAL_TIME
   char              type;             // DEAL_TYPE
   char              entry;            // DEAL_ENTRY
```

```
   string            symbol;           // DEAL_SYMBOL
   double            volume;           // DEAL_VOLUME
   double            price;            // DEAL_PRICE
   double            profit;           // DEAL_PROFIT
   double            swap;             // DEAL_SWAP
   double            commission;       // DEAL_COMMISSION
   long              magic;            // DEAL_MAGIC
   char              reason;           // DEAL_REASON
  };
//--- Structure to store the trade: the order of members corresponds to the position i
struct Trade
  {
   datetime          time_in;          // entry time
   ulong             ticket;           // position ID
   char              type;             // buy or sell
   double            volume;           // volume
   string            symbol;           // symbol
   double            price_in;         // entry price
   datetime          time_out;         // exit time
   double            price_out;        // exit price
   double            commission;       // entry and exit commission
   double            swap;             // swap
   double            profit;           // profit or loss
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_trades.sqlite"
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
```

```
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);
//--- add deals to the table
   if(!InsertDeals(db))
      return;
//--- show the first 10 deals
   Deal deals[], deal;
   ArrayResize(deals, 10);
   int request=DatabasePrepare(db, "SELECT * FROM DEALS");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   int i;
   for(i=0; DatabaseReadBind(request, deal); i++)
     {
      if(i>=10)
         break;
      deals[i].ticket=deal.ticket;
      deals[i].order_ticket=deal.order_ticket;
      deals[i].position_ticket=deal.position_ticket;
      deals[i].time=deal.time;
      deals[i].type=deal.type;
      deals[i].entry=deal.entry;
      deals[i].symbol=deal.symbol;
      deals[i].volume=deal.volume;
      deals[i].price=deal.price;
      deals[i].profit=deal.profit;
      deals[i].swap=deal.swap;
      deals[i].commission=deal.commission;
      deals[i].magic=deal.magic;
      deals[i].reason=deal.reason;
     }
//--- print the deals
   if(i>0)
     {
      ArrayResize(deals, i);
      PrintFormat("The first %d deals:", i);
      ArrayPrint(deals);
     }

//--- delete request after use
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accou
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
```

```
      //--- deals cannot be transformed to trades using a simple method through transa
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   ulong start=GetMicrosecondCount();
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,TICKET,TYPE,VOLUME,SYMBOL,PR
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the TRADES table failed with code ", GetLastError());
         return;
        }
   ulong transaction_time=GetMicrosecondCount()-start;

//--- show the first 10 deals
   Trade trades[], trade;
   ArrayResize(trades, 10);
   request=DatabasePrepare(db, "SELECT * FROM TRADES");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   for(i=0; DatabaseReadBind(request, trade); i++)
     {
```

```
      if(i>=10)
         break;
      trades[i].time_in=trade.time_in;
      trades[i].ticket=trade.ticket;
      trades[i].type=trade.type;
      trades[i].volume=trade.volume;
      trades[i].symbol=trade.symbol;
      trades[i].price_in=trade.price_in;
      trades[i].time_out=trade.time_out;
      trades[i].price_out=trade.price_out;
      trades[i].commission=trade.commission;
      trades[i].swap=trade.swap;
      trades[i].profit=trade.profit;
     }
//--- print trades
   if(i>0)
     {
      ArrayResize(trades, i);
      PrintFormat("\r\nThe first %d trades:", i);
      ArrayPrint(trades);
      PrintFormat("Filling the TRADES table took %.2f milliseconds",double(transactio
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2741
   The first 10 deals:
       [ticket] [order_ticket] [position_ticket]              [time] [type] [entry] [s
   [0] 34429573              0                 0 2019.09.05 22:39:59      2       0 ""
   [1] 34432127       51447238          51447238 2019.09.06 06:00:03      0       0 "U
   [2] 34432128       51447239          51447239 2019.09.06 06:00:03      1       0 "U
   [3] 34432450       51447565          51447565 2019.09.06 07:00:00      0       0 "E
   [4] 34432456       51447571          51447571 2019.09.06 07:00:00      1       0 "A
   [5] 34432879       51448053          51448053 2019.09.06 08:00:00      1       0 "U
   [6] 34432888       51448064          51448064 2019.09.06 08:00:00      0       0 "U
   [7] 34435147       51450470          51450470 2019.09.06 10:30:00      1       0 "E
   [8] 34435152       51450476          51450476 2019.09.06 10:30:00      0       0 "C
   [9] 34435154       51450479          51450479 2019.09.06 10:30:00      1       0 "E

   The first 10 trades:
                  [time_in] [ticket] [type] [volume] [symbol] [price_in]          [time
   [0] 2019.09.06 06:00:03 51447238      0  0.10000 "USDCAD"    1.32320 2019.09.06 18:
   [1] 2019.09.06 06:00:03 51447239      1  0.10000 "USDCHF"    0.98697 2019.09.06 18:
   [2] 2019.09.06 07:00:00 51447565      0  0.10000 "EURUSD"    1.10348 2019.09.09 03:
```

```
   [3] 2019.09.06 07:00:00 51447571      1  0.10000 "AUDUSD"    0.68203 2019.09.09 03:
   [4] 2019.09.06 08:00:00 51448053      1  0.10000 "USDCHF"    0.98701 2019.09.06 18:
   [5] 2019.09.06 08:00:00 51448064      0  0.10000 "USDJPY"  106.96200 2019.09.06 18:
   [6] 2019.09.06 10:30:00 51450470      1  0.10000 "EURUSD"    1.10399 2019.09.06 14:
   [7] 2019.09.06 10:30:00 51450476      0  0.10000 "GBPUSD"    1.23038 2019.09.06 14:
   [8] 2019.09.06 10:30:00 51450479      1  0.10000 "EURJPY"  118.12000 2019.09.06 14:
   [9] 2019.09.06 10:30:00 51450480      0  0.10000 "GBPJPY"  131.65300 2019.09.06 14:
   Filling the TRADES table took 12.51 milliseconds
*/
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create the DEALS table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
```

```
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table  with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
```

```
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tim
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code %d", __FUNCTION__, GetLastEr
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
```

```
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Guarda anche

DatabaseExecute, DatabaseFinalize

DatabaseReset

Reimposta una richiesta, come dopo aver chiamato DatabasePrepare().

```
int  DatabaseReset(
   int  request      // handle di richiesta ricevuto in DatabasePrepare
   );
```

Parametri

request

[in] L'handle della richiesta ottenuta in DatabasePrepare().

Valore di Ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - handle database non valido;
- Codici di errore SQLite che iniziano con ERR_DATABASE_ERROR (5601).

Nota

La funzione DatabaseReset() è destinata all'esecuzione multipla di una richiesta con valori di parametro diversi. Ad esempio, quando si aggiungono dati alla tabella in blocco utilizzando il comando INSERT, è necessario formare un set personalizzato di valori di campo per ogni voce.

A differenza di DatabasePrepare(), la chiamata DatabaseReset() non compila la stringa con i comandi SQL in una nuova richiesta, quindi DatabaseReset() viene eseguito molto più velocemente di DatabasePrepare().

DatabaseReset() viene utilizzato insieme alle funzioni DatabaseBind() e/o DatabaseBindArray() se i valori dei parametri di richiesta devono essere modificati dopo l'esecuzione DatabaseRead(). Ciò significa che prima di impostare nuovi valori dei parametri di richiesta (prima del blocco delle chiamate DatabaseBind/DatabaseBindArray), DatabaseReset() dovrebbe essere chiamato per resettarlo. La stessa richiesta parametrizzata deve essere creata utilizzando DatabasePrepare().

Proprio come DatabasePrepare(), DatabaseReset() non effettua una richiesta di database. Un'esecuzione della richiesta diretta viene eseguita durante la chiamata DatabaseRead() o DatabaseReadBind().

La chiamata DatabaseReset() non porta al ripristino dei valori dei parametri nella richiesta se sono stati impostati chiamando DatabaseBind()/DatabaseBindArray(), ovvero i parametri mantengono i loro valori. Pertanto, il valore di un solo parametro può essere modificato. Non è necessario impostare nuovamente tutti i parametri di richiesta dopo aver chiamato DatabaseReset().

Un handle di una richiesta rimosso utilizzando DatabaseFinalize() non può essere passato a DatabaseReset(). Ciò comporterà un errore.

Esempio:

```
//+------------------------------------------------------------------+
// | Funzione di avvio del programma Script                          |
//+------------------------------------------------------------------+
void OnStart()
```

```
  {
//--- crea o apre un database
   string filename="symbols.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " apertura fallita con codice ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " aperto con successo");
//--- se esiste la tabella SIMBOLI, eliminarla
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- elimina la tabella
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("Fallimento nell' eliminare la tabella SYMBOLS con codice ", GetLastErr
         DatabaseClose(db);
         return;
        }
     }
//--- crea la tabella SIMBOLI
   if(!DatabaseExecute(db, "CREATE TABLE SYMBOLS("
                       "NAME           TEXT    NOT NULL,"
                       "DESCRIPTION    TEXT            ,"
                       "PATH           TEXT            ,"
                       "SPREAD         INT             ,"
                       "POINT          REAL    NOT NULL,"
                       "DIGITS         INT     NOT NULL,"
                       "JSON           BLOB );"))
     {
      Print("DB: ", filename, " creazione tabella fallita con codice ", GetLastError()
      DatabaseClose(db);
      return;
     }
//--- visualizza l'elenco di tutti i campi nella tabella SYMBOLS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\") fallito, codice erro
      DatabaseClose(db);
      return;
     }

//--- crea una richiesta parametrizzata per aggiungere simboli alla tabella SYMBOLS
   string sql="INSERT INTO SYMBOLS (NAME,DESCRIPTION,PATH,SPREAD,POINT,DIGITS,JSON)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7);"; // request parameters
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
```

```
      {
       PrintFormat("DatabasePrepare() fallito con codice=%d", GetLastError());
       Print("SQL request: ", sql);
       DatabaseClose(db);
       return;
      }

//--- passa attraverso tutti i simboli e aggiungili alla tabella SYMBOLS
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- imposta i valori dei parametri prima di aggiungere un simbolo
      ResetLastError();
      string symbol=SymbolName(i, false);
      if(!DatabaseBind(request, 0, symbol))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__, Ge
         request_error=true;
         break;
        }
      //--- se la precedente chiamata DatabaseBind() ha avuto esito positivo, impostai
      if(!DatabaseBind(request, 1, SymbolInfoString(symbol, SYMBOL_DESCRIPTION)))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__, Ge
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, SymbolInfoString(symbol, SYMBOL_PATH)))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__, Ge
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 3, SymbolInfoInteger(symbol, SYMBOL_SPREAD)))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__, Ge
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 4, SymbolInfoDouble(symbol, SYMBOL_POINT)))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__, Ge
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 5, SymbolInfoInteger(symbol, SYMBOL_DIGITS)))
        {
```

```
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__, Ge
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 6, GetSymBolAsJson(symbol)))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__, Ge
         request_error=true;
         break;
        }

      //--- esegue una richiesta di inserimento della voce e verifica la presenza di u
      if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
        {
         PrintFormat("DatabaseRead() fallito con codice=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
      else
         PrintFormat("%d: aggiunti %s", i+1, symbol);
      //--- resetta la richiesta prima del prossimo aggiornamento dei parametri
      if(!DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() fallito con codice=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- fatto passando attraverso tutti i simboli

//--- stato delle transazioni
   if(request_error)
     {
      PrintFormat("Tabella SYMBOLS: fallimento nell'aggiungere %d simboli", symbols);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Tabella SYMBOLS: aggiunti %d simboli",symbols);
     }

//--- salva la tabella SYMBOLS in un file CSV
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
                     DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_QUOT
```

```
      Print("Database: tabella SYMBOLS salvata in ", csv_filename);
   else
      Print("Database: DatabaseExport(\"SELECT * FROM SYMBOLS\") fallita con codice",

//--- chiude il file del database e informa di ciò
   DatabaseClose(db);
   PrintFormat("Database: %s creato e chiuso", filename);
  }
//+------------------------------------------------------------------+
//| Restituisce una specifica di simbolo come JSON                   |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- indentazioni
   string indent1=Indent(1);
   string indent2=Indent(2);
   string indent3=Indent(3);
//---
   int digits=(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   string json="{"+
               "\n"+indent1+"\"ConfigSymbols\":["+
               "\n"+indent2+"{"+
               "\n"+indent3+"\"Symbol\":\""+symbol+"\","+
               "\n"+indent3+"\"Path\":\""+SymbolInfoString(symbol, SYMBOL_PATH)+"\","+
               "\n"+indent3+"\"CurrencyBase\":\""+SymbolInfoString(symbol, SYMBOL_CURR
               "\n"+indent3+"\"CurrencyProfit\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"CurrencyMargin\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"ColorBackground\":\""+ColorToString((color)SymbolInfoIn
               "\n"+indent3+"\"Digits\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"Point\":\""+DoubleToString(SymbolInfoDouble(symbol, SYM
               "\n"+indent3+"\"TickBookDepth\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ChartMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeCalcMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"OrderMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"CalculationMode\":\""+IntegerToString(SymbolInfoIntege
               "\n"+indent3+"\"ExecutionMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ExpirationMode\":\""+IntegerToString(SymbolInfoInteger
               "\n"+indent3+"\"FillFlags\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"ExpirFlags\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"Spread\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"TickValue\":\""+StringFormat("%G", (SymbolInfoDouble(sy
               "\n"+indent3+"\"TickSize\":\""+StringFormat("%G", (SymbolInfoDouble(sym
               "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",(SymbolInfoDouble
               "\n"+indent3+"\"StopsLevel\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",(SymbolInfoDouble(sym
               "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",(SymbolInfoDouble(sym
               "\n"+indent3+"\"VolumeStep\":\""+StringFormat("%G",(SymbolInfoDouble(sy
               "\n"+indent3+"\"VolumeLimit\":\""+StringFormat("%G",(SymbolInfoDouble(s
```

```
                "\n"+indent3+"\"SwapMode\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent3+"\"SwapLong\":\""+StringFormat("%G",(SymbolInfoDouble(symb
                "\n"+indent3+"\"SwapShort\":\""+StringFormat("%G",(SymbolInfoDouble(sy
                "\n"+indent3+"\"Swap3Day\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//| Forma un rientro fatto di spazi                                  |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
   return indent;
  }
/*
 Risultato:
 Database: symbols.sqlite aperto correttamente
  #| cid name        type notnull dflt_value pk
  -+-----------------------------------------
  1|   0 NAME        TEXT       1             0
  2|   1 DESCRIPTION TEXT       0             0
  3|   2 PATH        TEXT       0             0
  4|   3 SPREAD      INT        0             0
  5|   4 POINT       REAL       1             0
  6|   5 DIGITS      INT        1             0
  7|   6 JSON        BLOB       0             0
  1: added EURUSD
  2: added GBPUSD
  3: added USDCHF
  ...
  82: added USDCOP
  83: added USDARS
  84: added USDCLP
  Table SYMBOLS: added 84 symbols
  Database: table SYMBOLS saved in symbols.csv
  Database: symbols.sqlite created and closed
*/
```

Vedi anche

DatabasePrepare, DatabaseBind, DatabaseBindArray, DatabaseFinalize

DatabaseBind

Imposta un valore di parametro in una richiesta.

```
bool  DatabaseBind(
   int   request,      // l'handle di una richiesta creata in DatabasePrepare
   int   index,        // l'indice dei parametri nella richiesta
   T     value         // il valore di un parametro di tipo semplice
   );
```

Parametri

request

[in] L'handle di una richiesta creata in DatabasePrepare().

index

[in] L'indice dei parametri nella richiesta per cui deve essere impostato un valore. La numerazione inizia con zero.

value

[in] Il valore da impostare. Tipi estesi: bool, char, uchar, short, ushart, int, uint, color, datetime, long, ulong, float, double, string.

Valore di Ritorno

Restituisce true se ha esito positivo, altrimenti - false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INVALID_PARAMETER (4003) - tipo non supportato;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle database non valido;
- ERR_DATABASE_NOT_READY (5128) - al momento non è possibile utilizzare la funzione per effettuare una richiesta. La richiesta è in esecuzione o già completa. DatabaseReset() dovrebbe essere chiamato.

Nota

La funzione viene utilizzata nel caso in cui una richiesta SQL contenga i valori parametrizzabili "?" o "?N" dove N indica l'indice dei parametri (a partire da uno). Allo stesso tempo, l'indicizzazione dei parametri in DatabaseBind() inizia da zero.

Per esempio:

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

La funzione può essere chiamata immediatamente dopo la creazione di una richiesta parametrizzata DatabasePrepare() o dopo che la richiesta è stata ripristinata utilizzando DatabaseReset().

Utilizzare questa funzione insieme a DatabaseReset() per eseguire la richiesta tutte le volte necessarie con valori di parametro diversi.

La funzione è progettata per funzionare con semplici parametri di tipo. Se un parametro deve essere verificato su un array, utilizzare la funzione DatabaseBindArray().

Esempio:

```
//+------------------------------------------------------------------+
// | Funzione di avvio del programma Script                          |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- ricorda l'ora di inizio prima di ricevere i ticks
   uint start=GetTickCount();
//--- richiede la cronistoria dei tick al giorno
   ulong to=TimeCurrent()*1000;
   ulong from=to-PeriodSeconds(PERIOD_D1)*1000;
   if(CopyTicksRange(_Symbol, ticks, COPY_TICKS_ALL, from, to)==-1)
     {
      PrintFormat("%s: CopyTicksRange(%s - %s) fallito, errore=%d",
                  _Symbol, TimeToString(datetime(from/1000)), TimeToString(datetime(to
      return;
     }
   else
     {
      //--- quanti ticks sono stati ricevuti e quanto tempo ci è voluto per riceverli
      PrintFormat("%s: CopyTicksRange ha ricevuto %d ticks in %d ms (da %s a %s)",
                  _Symbol, ArraySize(ticks), GetTickCount()-start,
                  TimeToString(datetime(from/1000)), TimeToString(datetime(to/1000)));
     }

//--- imposta il nome del file per la memorizzazione del database
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" - "+TimeToString(da
   StringReplace(filename, ":", "."); // Il carattere ":" non è consentito nei nomi de
//--- apre/crea il database nella cartella terminale comune
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("Database: ", filename, " apri file con codice ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " aperto con successo");

//--- crea la tabella TICKS
   if(!DatabaseExecute(db, "CREATE TABLE TICKS("
                       "SYMBOL             CHAR(10),"
                       "TIME               INT NOT NULL,"
                       "BID                REAL,"
                       "ASK                REAL,"
                       "LAST               REAL,"
                       "VOLUME             INT,"
                       "TIME_MSC           INT,"
```

```
                        "VOLUME_REAL        REAL);"))
     {
      Print("DB: ", filename, " creazione tabella TICKS fallita con codice ", GetLastE
      DatabaseClose(db);
      return;
     }
//--- visualizza l'elenco di tutti i campi nella tabella TICKS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") fallito, codice errore=
      DatabaseClose(db);
      return;
     }
//--- crea una richiesta parametrizzata per aggiungere ticks alla tabella TICKS
   string sql="INSERT INTO TICKS (SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAL
              " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // parametri di richiesta
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() fallito con codice=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }
//--- imposta il valore del primo parametro di richiesta
   DatabaseBind(request, 0, _Symbol);
//--- ricorda l'ora di inizio prima di aggiungere ticks alla tabella TICKS
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
     {
      //--- imposta i valori dei parametri rimanenti prima di aggiungere la voce
      ResetLastError();
      if(!DatabaseBind(request, 1, ticks[i].time))
        {
         PrintFormat("DatabaseBind() fallito con codice=%d", GetLastError());
         PrintFormat("Tick #%d riga=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
      //--- se la precedente chiamata DatabaseBind() ha avuto esito positivo, imposta
      if(!request_error && !DatabaseBind(request, 2, ticks[i].bid))
        {
         PrintFormat("DatabaseBind() fallito con codice=%d", GetLastError());
         PrintFormat("Tick #%d riga=%d", i+1, __LINE__);
         request_error=true;
         break;
```

```
      }
   if(!request_error && !DatabaseBind(request, 3, ticks[i].ask))
      {
       PrintFormat("DatabaseBind() fallito con codice=%d", GetLastError());
       PrintFormat("Tick #%d riga=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 4, ticks[i].last))
      {
       PrintFormat("DatabaseBind() fallito con codice=%d", GetLastError());
       PrintFormat("Tick #%d riga=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 5, ticks[i].volume))
      {
       PrintFormat("DatabaseBind() fallito con codice=%d", GetLastError());
       PrintFormat("Tick #%d riga=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 6, ticks[i].time_msc))
      {
       PrintFormat("DatabaseBind() fallito con codice=%d", GetLastError());
       PrintFormat("Tick #%d riga=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
   if(!request_error && !DatabaseBind(request, 7, ticks[i].volume_real))
      {
       PrintFormat("DatabaseBind() fallito con codice=%d", GetLastError());
       PrintFormat("Tick #%d riga=%d", i+1, __LINE__);
       request_error=true;
       break;
      }

   //--- esegue una richiesta di inserimento della voce e verifica la presenza di u
   if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_
      {
       PrintFormat("DatabaseRead() fallito con codice=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
   //--- resetta la richiesta prima del prossimo aggiornamento dei parametri
   if(!request_error && !DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() fallito con codice=%d", GetLastError());
```

```
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- fatto passando attraverso tutti i ticks

//--- stato delle transazioni
   if(request_error)
     {
      PrintFormat("Tabella TICKS: fallita per aggiungere %d ticks ", ArraySize(ticks))
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Tabella TICKS: aggiunti %d ticks in %d ms",
                  ArraySize(ticks), GetTickCount()-start);
     }

//--- chiude il file del database e informa di ciò
   DatabaseClose(db);
   PrintFormat("Database: %s creato e chiuso", filename);
  }
/*
 Risultato:
  EURUSD: CopyTicksRange received 268061 ticks in 47 ms (from 2020.03.18 12:40 to 2020
 Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite aperto con successo
  #| cid name        type     notnull dflt_value pk
  -+-----------------------------------------------
  1|   0 SYMBOL      CHAR(10)       0              0
  2|   1 TIME        INT            1              0
  3|   2 BID         REAL           0              0
  4|   3 ASK         REAL           0              0
  5|   4 LAST        REAL           0              0
  6|   5 VOLUME      INT            0              0
  7|   6 TIME_MSC    INT            0              0
  8|   7 VOLUME_REAL REAL           0              0
  Table TICKS: added 268061 ticks in 797 ms
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite created and closed
  OnCalculateCorrelation=0.87 2020.03.19 13:00:  EURUSD vs GBPUSD  PERIOD_M30
*/
```

Vedi anche

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBindArray

DatabaseBindArray

Imposta un array come valore di parametro.

```
bool  DatabaseBind(
   int   request,       // l'handle di una richiesta creata in DatabasePrepare
   int   index,         // l'indice dei parametri nella richiesta
   T&    array[]        // valore del parametro come array
   );
```

Parametri

request

[in] L'handle di una richiesta creata in DatabasePrepare().

index

[in] L'indice dei parametri nella richiesta per cui deve essere impostato un valore. La numerazione inizia con zero.

array[]

[in] L'array da impostare come valore del parametro di richiesta.

Valore di Ritorno

Restituisce true se ha esito positivo, altrimenti - false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INVALID_PARAMETER (4003) - tipo non supportato;
- ERR_ARRAY_BAD_SIZE (4011) - la grandezza dell'array in byte supera INT_MAX;
- ERR_DATABASE_INVALID_HANDLE (5121) - handle database non valido;
- ERR_DATABASE_NOT_READY (5128) - al momento non è possibile utilizzare la funzione per effettuare una richiesta (la richiesta è in esecuzione o già completa, è necessario chiamare DatabaseReset).

Nota

La funzione viene utilizzata nel caso in cui una richiesta SQL contenga i valori parametrizzabili "?" o "?N" dove N indica l'indice dei parametri (a partire da uno). Allo stesso tempo, l'indicizzazione dei parametri in DatabaseBindArray() inizia da zero.

Per esempio:

```
INSERT INTO table VALUES (?,?,?)
```

La funzione può essere chiamata immediatamente dopo la creazione di una richiesta parametrizzata DatabasePrepare() o dopo che la richiesta è stata ripristinata utilizzando DatabaseReset().

Utilizzare questa funzione insieme a DatabaseReset() per eseguire la richiesta tutte le volte necessarie con valori di parametro diversi.

Esempio:

```
//+------------------------------------------------------------------+
// | Funzione di avvio del programma Script                          |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
//--- apre la finestra di dialogo per selezionare i file con l'estensione DAT
   string selected_files[];
   if(!FileSelectDialog("Seleziona i file da scaricare", NULL,
                        "Data files (*.dat)|*.dat|All files (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("File non selezionati. Exit");
      return;
     }
//--- ottiene la dimensione dei file
   ulong filesize[];
   int filehandle[];
   int files=ArraySize(selected_files);
   ArrayResize(filesize, files);
   ZeroMemory(filesize);
   ArrayResize(filehandle, files);
   double total_size=0;
   for(int i=0; i<files; i++)
     {
      filehandle[i]=FileOpen(selected_files[i], FILE_READ|FILE_BIN);
      if(filehandle[i]!=INVALID_HANDLE)
        {
         filesize[i]=FileSize(filehandle[i]);
         //PrintFormat("%d, %s handle=%d %d bytes", i, selected_files[i], filehandle[i
         total_size+=(double)filesize[i];
        }
     }
//--- controlla la grandezza comune dei file
   if(total_size==0)
     {
      PrintFormat("La grandezza totale dei file è 0. Exit");
      return;
     }

//--- crea o apre il database nella cartella comune del terminale
   string filename="dat_files.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " apertura fallita con codice ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " aperto con successo");
//--- se esiste la tabella FILES, eliminala
   if(DatabaseTableExists(db, "FILES"))
     {
```

```
      //--- elimina la tabella
      if(!DatabaseExecute(db, "DROP TABLE FILES"))
        {
         Print("Fallimento nell'eliminare la tabella FILES con codice ", GetLastError
         DatabaseClose(db);
         return;
        }
     }
//--- crea la tabella FILES
   if(!DatabaseExecute(db, "CREATE TABLE FILES("
                       "NAME           TEXT NOT NULL,"
                       "SIZE           INT  NOT NULL,"
                       "PERCENT_SIZE   REAL NOT NULL,"
                       "DATA           BLOB NOT NULL);"))
     {
      Print("DB: fallimento nel creare la tabella FILES con codice ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- visualizza l'elenco di tutti i campi nella tabella FILES
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") fallito, codice errore=
      DatabaseClose(db);
      return;
     }

//--- crea una richiesta parametrizzata per aggiungere file alla tabella FILES
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // richieste parametri
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() fallito con codice=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- passa attraverso tutti i file e li aggiunge alla tabella FILES
   bool request_error=false;
   DatabaseTransactionBegin(db);
   int count=0;
   uint size;
   for(int i=0; i<files; i++)
     {
      if(filehandle[i]!=INVALID_HANDLE)
        {
         char data[];
```

```
      size=FileReadArray(filehandle[i], data);
      if(size==0)
        {
         PrintFormat("FileReadArray(%s) fallita con codice %d", selected_files[i],
         continue;
        }

      count++;
      //--- imposta i valori dei parametri prima di aggiungere il file alla tabella
      if(!DatabaseBind(request, 0, selected_files[i]))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__,
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 1, size))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__,
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, double(size)*100./total_size))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__,
         request_error=true;
         break;
        }
      if(!DatabaseBindArray(request, 3, data))
        {
         PrintFormat("DatabaseBind() fallito alla riga %d con codice=%d", __LINE__,
         request_error=true;
         break;
        }
      //--- esegue una richiesta di inserimento della voce e verifica la presenza d
      if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
        {
         PrintFormat("DatabaseRead() fallito con codice=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
      else
         PrintFormat("%d. %s: %d byte", count, selected_files[i],size);
      //--- resetta la richiesta prima del prossimo aggiornamento dei parametri
      if(!DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() fallito con codice=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
```

```
            break;
           }
        }
     }
//--- stato delle transazioni
   if(request_error)
     {
      PrintFormat("Tabella FILES: fallita per aggiungere %d files", count);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Tabella FILES: aggiunti %d file", count);
     }

//--- chiude il file del database e informa di ciò
   DatabaseClose(db);
   PrintFormat("Database: %s creato e chiuso", filename);
  }
```

Vedi anche

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBind

DatabaseRead

Si sposta alla voce successiva a seguito di una richiesta.

```
bool  DatabaseRead(
   int  request      // handle della richiesta ricevuto in DatabasePrepare
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INVALID_PARAMETER (4003) - no table name specified (empty string or NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INTERNAL (5120) - internal database error;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_EXECUTE (5124) - request execution error;
- ERR_DATABASE_NO_MORE_DATA (5126) - no table exists (not an error, normal completion).

Guarda anche

DatabasePrepare, DatabaseReadBind

DatabaseReadBind

Passa al record successivo e legge i dati nella struttura da esso.

```
bool  DatabaseReadBind(
   int    request,           // l'handle di una richiesta creata in DatabasePrepare
   void&  struct_object      // il riferimento alla struttura per la lettura del reco
   );
```

Parametri

request

[in] L'handle di una richiesta creata in DatabasePrepare().

struct_object

[out] Il riferimento alla struttura in cui devono essere letti i dati del record corrente. La struttura dovrebbe avere solo tipi numerici e/o stringhe (le matrici non sono consentite) come membri e non può essere discendente.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INVALID_PARAMETER (4003) - no table name specified (empty string or NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INTERNAL (5120) - internal database error;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_EXECUTE (5124) - request execution error;
- ERR_DATABASE_NO_MORE_DATA (5126) - no table exists (not an error, normal completion).

Nota

Un numero di campi nella struttura *struct_object* non deve superare DatabaseColumnsCount(). Se il numero di campi nella struttura *struct_object* è inferiore al numero di campi nel record, viene eseguita la lettura parziale. I dati rimanenti possono essere ottenuti esplicitamente utilizzando le corrispondenti funzioni DatabaseColumnText(), DatabaseColumnInteger() ed altre funzioni.

Example:

```
struct Person
  {
   int               id;
   string            name;
   int               age;
   string            address;
   double            salary;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

```
  {
   int db;
   string filename="company.sqlite";
//--- open
   db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATABASE_
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the table COMPANY exists then drop the table
   if(DatabaseTableExists(db, "COMPANY"))
     {
      //--- delete the table
      if(!DatabaseExecute(db, "DROP TABLE COMPANY"))
        {
         Print("Failed to drop table COMPANY with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- create table
   if(!DatabaseExecute(db, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- insert data
   if(!DatabaseExecute(db, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
     {
      Print("DB: ", filename, " insert failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- prepare the request
   int request=DatabasePrepare(db, "SELECT * FROM COMPANY WHERE SALARY>15000");
   if(request==INVALID_HANDLE)
     {
```

```
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- print records
   Person person;
   Print("Persons with salary > 15000:");
   for(int i=0; DatabaseReadBind(request, person); i++)
      Print(i, ":  ", person.id, " ", person.name, " ", person.age, " ", person.addres
//--- delete request after use
   DatabaseFinalize(request);

   Print("Some statistics:");
//--- prepare new request about total salary
   request=DatabasePrepare(db, "SELECT SUM(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double total_salary;
      DatabaseColumnDouble(request, 0, total_salary);
      Print("Total salary=", total_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- prepare new request about average salary
   request=DatabasePrepare(db, "SELECT AVG(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      ResetLastError();
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double aver_salary;
      DatabaseColumnDouble(request, 0, aver_salary);
      Print("Average salary=", aver_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- close database
```

```
   DatabaseClose(db);
  }
//+------------------------------------------------------------------+
/*
Output:
Persons with salary > 15000:
0:  1 Paul 32 California 25000.0
1:  3 Teddy 23 Norway 20000.0
2:  4 Mark 25 Rich-Mond  65000.0
Some statistics:
Total salary=125000.0
Average salary=31250.0
*/
```

Guarda anche

DatabasePrepare, DatabaseRead

DatabaseFinalize

Rimuove una richiesta creata in DatabasePrepare ().

```
void  DatabaseFinalize(
   int  request      // handle della richiesta ricevuto in DatabasePrepare
   );
```

Parametri

request

[in] Handle della richiesta ricevuto in DatabasePrepare().

Valore di ritorno

Nessuno.

Nota

Se l'handle non è valido, la funzione imposta l'errore ERR_DATABASE_INVALID_HANDLE. Puoi controllare l'errore usando GetLastError().

Guarda anche

DatabasePrepare, DatabaseExecute

DatabaseTransactionBegin

Inizia l'esecuzione della transazione.

```
bool  DatabaseTransactionBegin(
   int  database      // handle database ricevuto in DatabaseOpen
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INTERNAL_ERROR (4001) - critical runtime error;
- ERR_INVALID_PARAMETER (4003) - sql parameter contains an empty string;
- ERR_NOT_ENOUGH_MEMORY (4004) - insufficient memory;
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INTERNAL (5120) - internal database error;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_EXECUTE (5124) - request execution error.

Nota

La funzione DatabaseTransactionBegin() deve essere chiamata prima dell'esecuzione di una transazione. Qualsiasi transazione dovrebbe iniziare con la chiamata a DatabaseTransactionBegin() e terminare con la chiamata a DatabaseTransactionCommit().

Example:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=AccountInfoString(ACCOUNT_SERVER) +"_"+IntegerToString(AccountInfo
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
```

```
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);

//--- measure the transaction execution speed using DatabaseTransactionBegin/DatabaseT
   ulong start=GetMicrosecondCount();
   bool fast_transactions=true;
   InsertDeals(db, fast_transactions);
   double fast_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit

//--- delete the DEALS table, and then create it again
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- create a new DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }

//--- test again, this time without using DatabaseTransactionBegin/DatabaseTransaction
   fast_transactions=false;
   start=GetMicrosecondCount();
   InsertDeals(db, fast_transactions);
   double slow_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit
//--- report gain in time
   PrintFormat("Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acc
//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2737
   Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit: time=48.5 r
```

```
   Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit: time=25818.
   Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acceleration by
*/
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop table with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create the table DEALS failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
```

```
bool InsertDeals(int database, bool begintransaction=true)
  {
//--- Auxiliary variables
   ulong    deal_ticket;          // deal ticket
   long     order_ticket;         // the ticket of the order by which the deal was exec
   long     position_ticket;      // ID of the position to which the deal belongs
   datetime time;                 // deal execution time
   long     type ;                // deal type
   long     entry ;               // deal direction
   string   symbol;               // the symbol fro which the deal was executed
   double   volume;               // operation volume
   double   price;                // price
   double   profit;               // financial result
   double   swap;                 // swap
   double   commission;           // commission
   long     magic;                // Magic number
   long     reason;               // deal execution reason or source
//--- go through all deals and add to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      // --- lock the database before executing transactions
      DatabaseTransactionBegin(database);
     }
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIN
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%dwith code %d", __FUNCTION__, deal_t
```

```
          PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
          failed=true;
          break;
         }
      }
//--- check for transaction execution errors
   if(failed)
     {
      //--- if fast transaction performance method is used
      if(begintransaction)
        {
         //--- roll back all transactions and unlock the database
         DatabaseTransactionRollback(database);
        }
      Print("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastError());
      return(false);
     }
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      //--- all transactions have been performed successfully - record changes and unl
      DatabaseTransactionCommit(database);
     }
//--- successful completion
   return(true);
  }
//+------------------------------------------------------------------+
```

Guarda anche

DatabaseExecute, DatabasePrepare, DatabaseTransactionCommit, DatabaseTransactionRollback

DatabaseTransactionCommit

Completa l'esecuzione della transazione.

```
bool  DatabaseTransactionCommit(
   int  database      // handle database ricevuto in DatabaseOpen
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INTERNAL_ERROR (4001) - critical runtime error;
- ERR_INVALID_PARAMETER (4003) - sql parameter contains an empty string;
- ERR_NOT_ENOUGH_MEMORY (4004) - insufficient memory;
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INTERNAL (5120) - internal database error;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_EXECUTE (5124) - request execution error.

Nota

La funzione DatabaseTransactionCommit() completa tutte le transazioni eseguite dopo aver chiamato la funzione DatabaseBeginTransaction(). Qualsiasi transazione dovrebbe iniziare con la chiamata a DatabaseTransactionBegin() e terminare con la chiamata a DatabaseTransactionCommit() per il completamento con esito positivo.

Guarda anche

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionRollback

DatabaseTransactionRollback

Ripristina le transazioni.

```
bool  DatabaseTransactionRollback(
   int  database      // handle database ricevuto in DatabaseOpen
   );
```

Parametri

database

[in] Handle database ricevuto in DatabaseOpen().

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_INTERNAL_ERROR (4001) - critical runtime error;
- ERR_INVALID_PARAMETER (4003) - sql parameter contains an empty string;
- ERR_NOT_ENOUGH_MEMORY (4004) - insufficient memory;
- ERR_WRONG_STRING_PARAMETER (5040) - error converting a request into a UTF-8 string;
- ERR_DATABASE_INTERNAL (5120) - internal database error;
- ERR_DATABASE_INVALID_HANDLE (5121) - invalid database handle;
- ERR_DATABASE_EXECUTE (5124) - request execution error.

Nota

La chiamata DatabaseTransactionRollback() annulla tutte le transazioni eseguite dopo aver chiamato la funzione DatabaseTransactionBegin(). La funzione DatabaseTransactionRollback() è necessaria per il rollback delle modifiche in un database nel caso in cui si verifichino errori durante l'esecuzione di una transazione.

Guarda anche

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionCommit

DatabaseColumnsCount

Ottiene il numero di campi in una richiesta.

```
int  DatabaseColumnsCount(
   int  request      // handle di richiesta ricevuto in DatabasePrepare
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

Valore di ritorno

Numero di campi o -1 in caso di errore. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle.

Nota

Non è necessario chiamare la funzione DatabaseRead() per ottenere il numero di campi di una richiesta creata in DatabasePrepare(). Per le restanti funzioni DatabaseColumnXXX(), DatabaseRead() deve essere chiamato in via preliminare.

Guarda anche

DatabasePrepare, DatabaseFinalize, DatabaseClose

DatabaseColumnName

Ottiene il nome campo per indice.

```
bool  DatabaseColumnName(
   int       request,     // handle di richiesta ricevuto in DatabasePrepare
   int       column,      // indice di campo nella richiesta
   string&   name         // il riferimento alla variabile per ricevere il nome del car
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

name

[out] Variabile per la scrittura del nome del campo.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

Guarda anche

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType

DatabaseColumnType

Ottiene un tipo di campo per indice.

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,     // handle di richiesta ricevuto in DatabasePrepare
   int  column       // indice di campo nella richiesta
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

Valore di ritorno

Restituisce il tipo di campo dall'enumerazione ENUM_DATABASE_FIELD_TYPE. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

ENUM_DATABASE_FIELD_TYPE

ID	Descrizione
DATABASE_FIELD_TYPE_INVALID	Errore durante l'acquisizione del tipo, il codice di errore può essere ottenuto utilizzando int GetLastError()
DATABASE_FIELD_TYPE_INTEGER	Tipo intero(integer)
DATABASE_FIELD_TYPE_FLOAT	Tipo reale(real)
DATABASE_FIELD_TYPE_TEXT	Tipo stringa(string)
DATABASE_FIELD_TYPE_BLOB	Tipo binario(binary)
DATABASE_FIELD_TYPE_NULL	Tipo speciale NULL

Guarda anche

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnName

DatabaseColumnSize

Ottiene la grandezza del campo in byte.

```
int  DatabaseColumnSize(
   int  request,     // handle di richiesta ricevuto in DatabasePrepare
   int  column       // indice di campo nella richiesta
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

Valore di ritorno

In caso di successo, viene restituita la grandezza del campo in byte, altrimenti -1. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

Guarda anche

DatabasePrepare, DatabaseColumnBlob, DatabaseColumnsCount, DatabaseColumnName, DatabaseColumnType

DatabaseColumnText

Ottiene un valore di campo come stringa dal record corrente.

```
bool  DatabaseColumnText(
   int      request,     // handle di richiesta ricevuto in DatabasePrepare
   int      column,      // indice di campo nella richiesta
   string&  value        // il riferimento alla variabile per ricevere il valore
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

value

[out] Riferimento alla variabile per la scrittura del valore del campo.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

Per leggere il valore dal record successivo, chiamare preliminarmente DatabaseRead().

Guarda anche

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnInteger

Ottiene il valore di tipo int dal record corrente.

```
bool  DatabaseColumnInteger(
   int   request,     // handle di richiesta ricevuto in DatabasePrepare
   int   column,      // indice di campo nella richiesta
   int&  value        // il riferimento alla variabile per ricevere il valore
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

value

[out] Riferimento alla variabile per la scrittura del valore del campo.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

Per leggere il valore dal record successivo, chiamare preliminarmente DatabaseRead().

Guarda anche

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnLong

Ottiene il valore di tipo long dal record corrente.

```
bool  DatabaseColumnLong(
   int    request,     // handle di richiesta ricevuto in DatabasePrepare
   int    column,      // indice di campo nella richiesta
   long&  value        // il riferimento alla variabile per ricevere il valore
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

value

[out] Riferimento alla variabile per la scrittura del valore del campo.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

Per leggere il valore dal record successivo, chiamare preliminarmente DatabaseRead().

Guarda anche

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnDouble

Ottiene il valore di tipo double dal record corrente.

```
bool  DatabaseColumnDouble(
   int      request,     // handle di richiesta ricevuto in DatabasePrepare
   int      column,      // indice di campo nella richiesta
   double&  value        // il riferimento alla variabile per ricevere il valore
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

value

[out] Riferimento alla variabile per la scrittura del valore del campo.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

Per leggere il valore dal record successivo, chiamare preliminarmente DatabaseRead().

Guarda anche

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnBlob

Ottiene un valore di campo come un array dal record corrente.

```
bool  DatabaseColumnBlob(
   int     request,      // handle di richiesta ricevuto in DatabasePrepare
   int     column,       // indice di campo nella richiesta
   void&   data[]        // il riferimento alla variabile per ricevere il valore
   );
```

Parametri

request

[in] Handle di richiesta ricevuto in DatabasePrepare().

column

[in] Indice di campo nella richiesta. La numerazione dei campi inizia da zero e non può superare DatabaseColumnsCount() - 1.

data[]

[out] Riferimento all'array per la scrittura del valore del campo.

Valore di ritorno

Restituisce true se ha esito positivo, altrimenti false. Per ottenere il codice di errore, utilizzare GetLastError(), le possibili risposte sono:

- ERR_DATABASE_INVALID_HANDLE (5121) - invalid request handle;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' index exceeds DatabaseColumnsCount() -1.

Nota

Il valore può essere ottenuto solo se almeno una chiamata DatabaseRead() è stata preliminarmente effettuata per "richiesta".

Per leggere il valore dal record successivo, chiamare preliminarmente DatabaseRead().

Guarda anche

DatabasePrepare, DatabaseColumnSize, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

Lavorare con DirectX

Le funzioni e gli shader di DirectX 11 sono progettati per la visualizzazione 3D direttamente sul chart dei prezzi.

La creazione di grafica 3D richiede un contesto grafico (DXContextCreate) con le dimensioni dell'immagine necessarie. Inoltre, è necessario preparare i buffer vertex ed index (DXBufferCreate), oltre a creare vertex shader e pixel shader (DXShaderCreate). Questo è sufficiente per visualizzare la grafica a colori.

Il successivo livello di grafica richiede gli input (DXInputSet) per passare parametri di rendering aggiuntivi agli shader. Ciò consente di impostare le posizioni della fotocamera e degli oggetti 3D, descrivere le fonti di luce e implementare il controllo del mouse e della tastiera.

Pertanto, le funzioni MQL5 integrate consentono di creare grafici 3D animati direttamente in MetaTrader 5 senza la necessità di strumenti di terze parti. Una scheda video dovrebbe supportare DX 11 e Shader Model 5.0 affinché le funzioni girino.

Per iniziare a lavorare con la libreria, leggi semplicemente l'articolo Come creare grafica 3D usando DirectX in MetaTrader 5.

Funzione	Azione
DXContextCreate	Crea un contesto grafico per il rendering di frame di una grandezza specificata
DXContextSetSize	Modifica le dimensioni di un frame di un contesto grafico creato in DXContextCreate()
DXContextSetSize	Ottiene una dimensione del frame di un contesto grafico creato in DXContextCreate()
DXContextClearColors	Imposta un colore specificato su tutti i pixel per il buffer di rendering
DXContextClearDepth	Cancella il buffer di profondità
DXContextGetColors	Ottiene un'immagine di dimensioni specificate ed offset da un contesto grafico
DXContextGetDepth	Ottiene il buffer di profondità di un fotogramma renderizzato
DXBufferCreate	Crea un buffer di un tipo specificato basato su un array di dati
DXTextureCreate	Crea una trama 2D da un rettangolo di una dimensione specificata tagliato da un'immagine passata
DXInputCreate	Crea input shader
DXInputSet	Imposta gli input dello shader
DXShaderCreate	Crea uno shader di un tipo specificato
DXShaderSetLayout	Imposta il layout dei vertici per il vertex shader
DXShaderInputsSet	Imposta gli input dello shader

Funzione	Azione
DXShaderTexturesSet	Imposta le texture dello shader
DXDraw	Esegue il rendering dei vertici del vertex buffer impostato in DXBufferSet()
DXDrawIndexed	Esegue il rendering delle primitive grafiche descritte dall' index buffer da DXBufferSet()
DXPrimiveTopologySet	Imposta il tipo di primitive per il rendering utilizzando DXDrawIndexed()
DXBufferSet	Imposta un buffer per il rendering corrente
DXShaderSet	Imposta uno shader per il rendering
DXHandleType	Restituisce un tipo di handle
DXRelease	Rilascia un handle

DXContextCreate

Crea un contesto grafico per il rendering di fotogrammi di una grandezza specificata.

```
int  DXContextCreate(
   uint  width,      // larghezza in pixel
   uint  height      // altezza in pixel
   );
```

Parametri

width

[in] Larghezza della cornice in pixel.

height

[in] Altezza della cornice in pixel.

Valore di Ritorno

Un handle per un contesto creato oppure INVALID_HANDLE in caso di errore. Per ricevere un codice di errore, dovrebbe essere chiamata la funzione GetLastError().

Nota

Tutti gli oggetti grafici creati con le funzioni DXBufferCreate, DXInputCreate, DXShaderCreate e DXTextureCreate possono essere utilizzati solo nel contesto grafico in cui sono stati creati.

La grandezza del frame può essere successivamente modificata inDXContextSetSize().

Un handle creato che non è più in uso dovrebbe essere esplicitamente rilasciato dalla funzione DXRelease().

DXContextSetSize

Modifica la grandezza di un frame di un contesto grafico creato in DXContextCreate().

```
bool  DXContextSetSize(
   int    context,      // handle del contesto grafico
   uint&  width,        // larghezza in pixel
   uint&  height        // altezza in pixel
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

width

[in] Larghezza della cornice in pixel.

height

[in] Altezza della cornice in pixel.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

La grandezza di un fotogramma di un contesto grafico deve essere modificata solo tra i rendering dei fotogrammi.

DXContextGetSize

Ottiene la grandezza del frame di un contesto grafico creato in DXContextCreate().

```
bool  DXContextGetSize(
   int    context,      // handle del contesto grafico
   uint&  width,        // larghezza in pixel
   uint&  height        // altezza in pixel
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

width

[out] Larghezza della cornice in pixel.

height

[out] Altezza della cornice in pixel.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

DXContextClearColors

Imposta un colore specificato su tutti i pixel per il buffer del rendering.

```
bool  DXContextClearColors(
   int              context,      // handle del contesto grafico
   const DXVector&  color         // colore
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

color

[in] Colore di rendering.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

La funzione DXContextClearColors() può essere utilizzata per cancellare il buffer del colore (color buffer) prima di eseguire il rendering del fotogramma successivo.

DXContextClearDepth

Cancella il buffer di profondità.

```
bool  DXContextClearDepth(
   int  context      // handle del contesto grafico
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

La funzione DXContextClearDepth() può essere utilizzata per cancellare il buffer della profondità prima di eseguire il rendering del fotogramma successivo.

DXContextGetColors

Ottiene un'immagine di grandezza ed offset specificati da un contesto grafico.

```
bool  DXContextGetColors(
   int    context,                      // handle del contesto grafico
   uint&  image[],                      // array di pixel dell'immagine
   int    image_width=WHOLE_ARRAY,      // larghezza dell'immagine in pixel
   int    image_height=WHOLE_ARRAY,     // altezza dell'immagine in pixel
   int    image_offset_x=0,             // offset X
   int    image_offset_y=0              // offset Y
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

image

[out] L'array di pixel *image_width*image_height* in formato ARGB.

image_width=WHOLE_ARRAY

[in] Image width in pixels.

image_height=WHOLE_ARRAY

[in] Image height in pixels.

image_offset_x=0

[in] X offset.

image_offset_y=0

[in] Y offset.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

DXContextGetDepth

Ottiene il buffer di profondità di un fotogramma renderizzato.

```
bool  DXContextGetDepth(
   int     context,      // handle del contesto grafico
   float&  image[]       // array di valori della profondità
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

image

[out] Array dei valori del buffer di profondità del frame renderizzato.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

Il buffer restituito contiene la profondità di ciascun pixel di un fotogramma renderizzato che può essere ottenuto in DXContextGetColors() in unità relative (da 0.0 ad 1.0).

DXBufferCreate

Crea un buffer di un tipo specificato basato su un array di dati.

```
int  DXBufferCreate(
   int                  context,              // handle del contesto grafico
   ENUM_DX_BUFFER_TYPE  buffer_type,          // tipo di buffer creato
   const void&          data[],               // dati buffer
   uint                 start=0,              // indice iniziale
   uint                 count=WHOLE_ARRAY     // numero di elementi
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

buffer_type

[in] Tipo di buffer dall'enumerazione ENUM_DX_BUFFER_TYPE.

data[]

[in] Dati per la creazione di un buffer.

start

[in] Indice del primo elemento dell'array, a partire dal quale vengono utilizzati i valori dell'array per creare un buffer. Per impostazione predefinita, i dati sono presi dall'inizio dell'array.

count

[in] Numero di valori. Per default, viene utilizzato l'intero array (count=WHOLE_ARRAY).

Valore di Ritorno

L'handle per un buffer creato o INVALID_HANDLE in caso di errore. Per ricevere un codice di errore, dovrebbe essere chiamata la funzione GetLastError().

Nota

Per l'inxed buffer, l'array *dati[]* dovrebbe essere di tipo 'uint', mentre il vertex buffer riceve l'array di strutture che descrivono i vertici.

Un handle creato che non è più in uso dovrebbe essere esplicitamente rilasciato dalla funzione DXRelease().

ENUM_DX_BUFFER_TYPE

ID	Valore	Descrizione
DX_BUFFER_VERTEX	1	Vertex buffer
DX_BUFFER_INDEX	2	Index buffer

DXTextureCreate

Crea una texture 2D da un rettangolo di una grandezza specificata tagliato da un'immagine che è stata passata.

```
int  DXTextureCreate(
   int             context,         // handle del contesto grafico
   ENUM_DX_FORMAT  format,          // formato colore del pixel
   uint            width,           // larghezza dell'immagine sorgente
   uint            height,          // altezza dell'immagine sorgente
   const void&     data[],          // array dei pixel dell'immagine sorgente
   uint            data_x,          // coordinata X del rettangolo utilizzato per crea
   uint            data_y,          // coordinata Y del rettangolo utilizzato per crea
   uint            data_width,      // larghezza rettangolo per creare la texture
   uint            data_height      // altezza rettangolo per creare la texture
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

format

[in] Formato colore pixel impostato dall'enumerazione ENUM_DX_FORMAT.

width

[in] Larghezza dell"immagine su cui si basa la texture.

height

[in] Altezza dell"immagine su cui si basa la texture.

data

[in] Array pixel dell'immagine su cui si basa la texture.

data_x

[in] coordinata X di un rettangolo (offset X) utilizzato per creare la texture.

data_y

[in] coordinata Y di un rettangolo (offset Y) utilizzato per creare la texture.

data_width

[in] Larghezza di un rettangolo utilizzato per creare la texture.

data_height

[in] Altezza del rettangolo utilizzato per creare la texture.

Valore di Ritorno

Handle texture oppure INVALID_HANDLE in caso di errore. Per ricevere un codice di errore, dovrebbe essere chiamata la funzione GetLastError().

Nota

Un handle creato che non è più in uso dovrebbe essere esplicitamente rilasciato dalla funzione DXRelease().

ENUM_DX_FORMAT

ID	Valore	Match in DXGI_FORMAT
DX_FORMAT_UNKNOWN	0	DXGI_FORMAT_UNKNOWN
DX_FORMAT_R32G32B32A32_TYPELESS	1	DXGI_FORMAT_R32G32B32A32_TYPELESS
DX_FORMAT_R32G32B32A32_FLOAT	2	DXGI_FORMAT_R32G32B32A32_FLOAT
DX_FORMAT_R32G32B32A32_UINT	3	DXGI_FORMAT_R32G32B32A32_UINT
DX_FORMAT_R32G32B32A32_SINT	4	DXGI_FORMAT_R32G32B32A32_SINT
DX_FORMAT_R32G32B32_TYPELESS	5	DXGI_FORMAT_R32G32B32_TYPELESS
DX_FORMAT_R32G32B32_FLOAT	6	DXGI_FORMAT_R32G32B32_FLOAT
DX_FORMAT_R32G32B32_UINT	7	DXGI_FORMAT_R32G32B32_UINT
DX_FORMAT_R32G32B32_SINT	8	DXGI_FORMAT_R32G32B32_SINT
DX_FORMAT_R16G16B16A16_TYPELESS	9	DXGI_FORMAT_R16G16B16A16_TYPELESS
DX_FORMAT_R16G16B16A16_FLOAT	10	DXGI_FORMAT_R16G16B16A16_FLOAT
DX_FORMAT_R16G16B16A16_UNORM	11	DXGI_FORMAT_R16G16B16A16_UNORM
DX_FORMAT_R16G16B16A16_UINT	12	DXGI_FORMAT_R16G16B16A16_UINT
DX_FORMAT_R16G16B16A16_SNORM	13	DXGI_FORMAT_R16G16B16A16_SNORM
DX_FORMAT_R16G16B16A16_SINT	14	DXGI_FORMAT_R16G16B16A16_SINT
DX_FORMAT_R32G32_TYPELESS	15	DXGI_FORMAT_R32G32_TYPELESS
DX_FORMAT_R32G32_FLOAT	16	DXGI_FORMAT_R32G32_FLOAT
DX_FORMAT_R32G32_UINT	17	DXGI_FORMAT_R32G32_UINT
DX_FORMAT_R32G32_SINT	18	DXGI_FORMAT_R32G32_SINT
DX_FORMAT_R32G8X24_TYPELESS	19	DXGI_FORMAT_R32G8X24_TYPELESS
DX_FORMAT_D32_FLOAT_S8X24_UINT	20	DXGI_FORMAT_D32_FLOAT_S8X24_UINT
DX_FORMAT_R32_FLOAT_X8X24_TYPELESS	21	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS
DX_FORMAT_X32_TYPELESS_G8X24_UINT	22	DXGI_FORMAT_X32_TYPELESS_G8X24_UINT

ID	Valore	Match in DXGI_FORMAT
DX_FORMAT_R10G10B10A2_TYPELESS	23	DXGI_FORMAT_R10G10B10A2_TYPELESS
DX_FORMAT_R10G10B10A2_UNORM	24	DXGI_FORMAT_R10G10B10A2_UNORM
DX_FORMAT_R10G10B10A2_UINT	25	DXGI_FORMAT_R10G10B10A2_UINT
DX_FORMAT_R11G11B10_FLOAT	26	DXGI_FORMAT_R11G11B10_FLOAT
DX_FORMAT_R8G8B8A8_TYPELESS	27	DXGI_FORMAT_R8G8B8A8_TYPELESS
DX_FORMAT_R8G8B8A8_UNORM	28	DXGI_FORMAT_R8G8B8A8_UNORM
DX_FORMAT_R8G8B8A8_UNORM_SRGB	29	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
DX_FORMAT_R8G8B8A8_UINT	30	DXGI_FORMAT_R8G8B8A8_UINT
DX_FORMAT_R8G8B8A8_SNORM	31	DXGI_FORMAT_R8G8B8A8_SNORM
DX_FORMAT_R8G8B8A8_SINT	32	DXGI_FORMAT_R8G8B8A8_SINT
DX_FORMAT_R16G16_TYPELESS	33	DXGI_FORMAT_R16G16_TYPELESS
DX_FORMAT_R16G16_FLOAT	34	DXGI_FORMAT_R16G16_FLOAT
DX_FORMAT_R16G16_UNORM	35	DXGI_FORMAT_R16G16_UNORM
DX_FORMAT_R16G16_UINT	36	DXGI_FORMAT_R16G16_UINT
DX_FORMAT_R16G16_SNORM	37	DXGI_FORMAT_R16G16_SNORM
DX_FORMAT_R16G16_SINT	38	DXGI_FORMAT_R16G16_SINT
DX_FORMAT_R32_TYPELESS	39	DXGI_FORMAT_R32_TYPELESS
DX_FORMAT_D32_FLOAT	40	DXGI_FORMAT_D32_FLOAT
DX_FORMAT_R32_FLOAT	41	DXGI_FORMAT_R32_FLOAT
DX_FORMAT_R32_UINT	42	DXGI_FORMAT_R32_UINT
DX_FORMAT_R32_SINT	43	DXGI_FORMAT_R32_SINT
DX_FORMAT_R24G8_TYPELESS	44	DXGI_FORMAT_R24G8_TYPELESS
DX_FORMAT_D24_UNORM_S8_UINT	45	DXGI_FORMAT_D24_UNORM_S8_UINT
DX_FORMAT_R24_UNORM_X8_TYPELESS	46	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DX_FORMAT_X24_TYPELESS_G8_UINT	47	DXGI_FORMAT_X24_TYPELESS_G8_UINT
DX_FORMAT_R8G8_TYPELESS	48	DXGI_FORMAT_R8G8_TYPELESS
DX_FORMAT_R8G8_UNORM	49	DXGI_FORMAT_R8G8_UNORM
DX_FORMAT_R8G8_UINT	50	DXGI_FORMAT_R8G8_UINT

ID	Valore	Match in DXGI_FORMAT
DX_FORMAT_R8G8_SNORM	51	DXGI_FORMAT_R8G8_SNORM
DX_FORMAT_R8G8_SINT	52	DXGI_FORMAT_R8G8_SINT
DX_FORMAT_R16_TYPELESS	53	DXGI_FORMAT_R16_TYPELESS
DX_FORMAT_R16_FLOAT	54	DXGI_FORMAT_R16_FLOAT
DX_FORMAT_D16_UNORM	55	DXGI_FORMAT_D16_UNORM
DX_FORMAT_R16_UNORM	56	DXGI_FORMAT_R16_UNORM
DX_FORMAT_R16_UINT	57	DXGI_FORMAT_R16_UINT
DX_FORMAT_R16_SNORM	58	DXGI_FORMAT_R16_SNORM
DX_FORMAT_R16_SINT	59	DXGI_FORMAT_R16_SINT
DX_FORMAT_R8_TYPELESS	60	DXGI_FORMAT_R8_TYPELESS
DX_FORMAT_R8_UNORM	61	DXGI_FORMAT_R8_UNORM
DX_FORMAT_R8_UINT	62	DXGI_FORMAT_R8_UINT
DX_FORMAT_R8_SNORM	63	DXGI_FORMAT_R8_SNORM
DX_FORMAT_R8_SINT	64	DXGI_FORMAT_R8_SINT
DX_FORMAT_A8_UNORM	65	DXGI_FORMAT_A8_UNORM
DX_FORMAT_R1_UNORM	66	DXGI_FORMAT_R1_UNORM
DX_FORMAT_R9G9B9E5_SHAREDEXP	67	DXGI_FORMAT_R9G9B9E5_SHAREDEXP
DX_FORMAT_R8G8_B8G8_UNORM	68	DXGI_FORMAT_R8G8_B8G8_UNORM
DX_FORMAT_G8R8_G8B8_UNORM	69	DXGI_FORMAT_G8R8_G8B8_UNORM
DX_FORMAT_BC1_TYPELESS	70	DXGI_FORMAT_BC1_TYPELESS
DX_FORMAT_BC1_UNORM	71	DXGI_FORMAT_BC1_UNORM
DX_FORMAT_BC1_UNORM_SRGB	72	DXGI_FORMAT_BC1_UNORM_SRGB
DX_FORMAT_BC2_TYPELESS	73	DXGI_FORMAT_BC2_TYPELESS
DX_FORMAT_BC2_UNORM	74	DXGI_FORMAT_BC2_UNORM
DX_FORMAT_BC2_UNORM_SRGB	75	DXGI_FORMAT_BC2_UNORM_SRGB
DX_FORMAT_BC3_TYPELESS	76	DXGI_FORMAT_BC3_TYPELESS
DX_FORMAT_BC3_UNORM	77	DXGI_FORMAT_BC3_UNORM
DX_FORMAT_BC3_UNORM_SRGB	78	DXGI_FORMAT_BC3_UNORM_SRGB
DX_FORMAT_BC4_TYPELESS	79	DXGI_FORMAT_BC4_TYPELESS
DX_FORMAT_BC4_UNORM	80	DXGI_FORMAT_BC4_UNORM
DX_FORMAT_BC4_SNORM	81	DXGI_FORMAT_BC4_SNORM

ID	Valore	Match in DXGI_FORMAT
DX_FORMAT_BC5_TYPELESS	82	DXGI_FORMAT_BC5_TYPELESS
DX_FORMAT_BC5_UNORM	83	DXGI_FORMAT_BC5_UNORM
DX_FORMAT_BC5_SNORM	84	DXGI_FORMAT_BC5_SNORM
DX_FORMAT_B5G6R5_UNORM	85	DXGI_FORMAT_B5G6R5_UNORM
DX_FORMAT_B5G5R5A1_UNORM	86	DXGI_FORMAT_B5G5R5A1_UNORM
DX_FORMAT_B8G8R8A8_UNORM	87	DXGI_FORMAT_B8G8R8A8_UNORM
DX_FORMAT_B8G8R8X8_UNORM	88	DXGI_FORMAT_B8G8R8X8_UNORM
DX_FORMAT_R10G10B10_XR_BIAS_A2_UNORM	89	DXGI_FORMAT_R10G10B10_XR_BIAS_A2_UNORM
DX_FORMAT_B8G8R8A8_TYPELESS	90	DXGI_FORMAT_B8G8R8A8_TYPELESS
DX_FORMAT_B8G8R8A8_UNORM_SRGB	91	DXGI_FORMAT_B8G8R8A8_UNORM_SRGB
DX_FORMAT_B8G8R8X8_TYPELESS	92	DXGI_FORMAT_B8G8R8X8_TYPELESS
DX_FORMAT_B8G8R8X8_UNORM_SRGB	93	DXGI_FORMAT_B8G8R8X8_UNORM_SRGB
DX_FORMAT_BC6H_TYPELESS	94	DXGI_FORMAT_BC6H_TYPELESS
DX_FORMAT_BC6H_UF16	95	DXGI_FORMAT_BC6H_UF16
DX_FORMAT_BC6H_SF16	96	DXGI_FORMAT_BC6H_SF16
DX_FORMAT_BC7_TYPELESS	97	DXGI_FORMAT_BC7_TYPELESS
DX_FORMAT_BC7_UNORM	98	DXGI_FORMAT_BC7_UNORM
DX_FORMAT_BC7_UNORM_SRGB	99	DXGI_FORMAT_BC7_UNORM_SRGB
DX_FORMAT_AYUV	100	DXGI_FORMAT_AYUV
DX_FORMAT_Y410	101	DXGI_FORMAT_Y410
DX_FORMAT_Y416	102	DXGI_FORMAT_Y416
DX_FORMAT_NV12	103	DXGI_FORMAT_NV12
DX_FORMAT_P010	104	DXGI_FORMAT_P010
DX_FORMAT_P016	105	DXGI_FORMAT_P016
DX_FORMAT_420_OPAQUE	106	DXGI_FORMAT_420_OPAQUE
DX_FORMAT_YUY2	107	DXGI_FORMAT_YUY2
DX_FORMAT_Y210	108	DXGI_FORMAT_Y210
DX_FORMAT_Y216	109	DXGI_FORMAT_Y216
DX_FORMAT_NV11	110	DXGI_FORMAT_NV11

ID	Valore	Match in DXGI_FORMAT
DX_FORMAT_AI44	111	DXGI_FORMAT_AI44
DX_FORMAT_IA44	112	DXGI_FORMAT_IA44
DX_FORMAT_P8	113	DXGI_FORMAT_P8
DX_FORMAT_A8P8	114	DXGI_FORMAT_A8P8
DX_FORMAT_B4G4R4A4_UNORM	115	DXGI_FORMAT_B4G4R4A4_UNORM
DX_FORMAT_P208	130	DXGI_FORMAT_P208
DX_FORMAT_V208	131	DXGI_FORMAT_V208
DX_FORMAT_V408	132	DXGI_FORMAT_V408
DX_FORMAT_FORCE_UINT	0xffffffff	DXGI_FORMAT_FORCE_UINT

DXInputCreate

Crea input shader.

```
int  DXInputCreate(
   int   context,        // handle del contesto grafico
   uint  input_size      // grandezza degli input in byte
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

input_size

[in] Grandezza della struttura dei parametri in byte.

Valore di Ritorno

L'handle per input shader o INVALID_HANDLE in caso di errore. Per ricevere un codice di errore, dovrebbe essere chiamata la funzione GetLastError().

Un handle creato che non è più in uso dovrebbe essere esplicitamente rilasciato dalla funzione DXRelease().

DXInputSet

Imposta gli input dello shader.

```
bool  DXInputSet(
   int           input,       // handle del contesto grafico
   const void&   data         // dati per l'impostazione
   );
```

Parametri

input

[in] Gestore di input per uno shader ottenuto in DXInputCreate().

data

[in] Dati per l'impostazione degli input dello shader.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

DXShaderCreate

Crea uno shader del tipo specificato.

```
int  DXShaderCreate(
   int                  context,           // handle del contesto grafico
   ENUM_DX_SHADER_TYPE  shader_type,       // tipo di shader
   const string         source,            // codice sorgente dello shader
   const string         entry_point,       // punto d'entrata
   string&              compile_error      // stringa per la ricezione di messaggi del
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

shader_type

[out] Il valore dell' enumerazione ENUM_DX_SHADER_TYPE.

source

[in] Codice sorgente shader in HLSL 5.

entry_point

[in] Punto di ingresso - nome della funzione nel codice sorgente.

compile_error

[in] Stringa per la ricezione di errori di compilazione.

Valore di Ritorno

Handle per lo shader o INVALID_HANDLE in caso di errore. Per ricevere un codice di errore, dovrebbe essere chiamata la funzione GetLastError().

Nota

Un handle creato che non è più in uso dovrebbe essere esplicitamente rilasciato dalla funzione DXRelease().

ENUM_DX_SHADER_TYPE

ID	Valore	Descrizione
DX_SHADER_VERTEX	0	Vertex shader
DX_SHADER_GEOMETRY	1	Geometry shader
DX_SHADER_PIXEL	2	Pixel shader

DXShaderSetLayout

Imposta il layout dei vertici per il vertex shader.

```
bool  DXShaderSetLayout(
int                        Shader,        // handle dello shader
   const DXVertexLayout&   layout[]       //
   );
```

Parametri

shader

[in] Handle di un vertex shader creato in DXShaderCreate().

layout[]

[in] Array della descrizione dei campi vertex. La descrizione è impostata dalla struttura DXVertexLayout:

```
struct DXVertexLayout
  {
   string          semantic_name;        // La semantica HLSL associata a questo ele
   uint            semantic_index;       // L'indice semantico per l'elemento. L' in
ENUM_DX_FORMATformato;// Il tipo di dati dei dati dell'elemento.
  };
```

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

Il layout deve corrispondere al tipo di vertici in un vertex buffer specificato. Deve inoltre corrispondere al tipo di input dei vertici utilizzato nel punto di ingresso nel codice del vertex shader.

Il vertex buffer per uno shader è impostato in DXBufferSet().

La struttura DXVertexLayout è una versione della struttura MSDN D3D11_INPUT_ELEMENT_DESC.

DXShaderInputsSet

Imposta gli input dello shader.

```
bool  DXShaderInputsSet(
   int          shader,        // handle shader
   const int&   inputs[]       // array degli handle degli input
   );
```

Parametri

shader

[in] Handle di uno shader creato in DXShaderCreate().

inputs[]

[in] Array degli handle di input creati utilizzando DXInputCreate().

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

La grandezza del parametro di input deve essere uguale al numero di oggetti cbuffer dichiarati nel codice dello shader.

DXShaderTexturesSet

Imposta le texture dello shader.

```
bool  DXShaderTexturesSet(
   int          shader,          // handle dello shader
   const  int&  textures[]       // array degli handle della struttura
   );
```

Parametri

shader

[in] Handle di uno shader creato in DXShaderCreate().

textures[]

[in] Array degli handle texture creati utilizzando DXTextureCreate().

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

La grandezza dell'array di texture dovrebbe essere uguale al numero di oggetti Texture2D dichiarati nel codice shader.

DXDraw

Esegue il rendering dei vertici del vertex buffer impostato DXBufferSet().

```
bool  DXDraw(
   int   context,               // handle del contesto grafico
   uint  start=0,               // indice del primo vertice
   uint  count=WHOLE_ARRAY      // numero di vertici
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

start

[in] Indice del primo vertice per il rendering.

count

[in] Numero di vertici da renderizzare.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

Gli shader devono essere impostati preliminarmente usando DXShaderSet() per il rendering dei vertici.

DXDrawIndexed

Esegue il rendering delle primitive grafiche descritte dall'index buffer da DXBufferSet().

```
bool  DXDrawIndexed(
   int   context,               // handle del contesto grafico
   uint  start=0,               // indice prima primitiva
   uint  count=WHOLE_ARRAY      // numero di primitive
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

start

[in] Indice della prima primitiva per il rendering.

count

[in] Numero di primitive per il rendering.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

Il tipo di primitive descritto dall' index buffer viene impostato utilizzando DXPrimiveTopologySet().

Il vertex buffer in DXBufferSet() dovrebbe essere impostato preliminarmente per il rendering delle primitive.

Inoltre, gli shader devono essere impostati preliminarmente usando DXShaderSet().

DXPrimiveTopologySet

Imposta il tipo di primitive per il rendering utilizzando DXDrawIndexed().

```
bool  DXPrimiveTopologySet(
   int                          context,               // handle del contesto grafico
   ENUM_DX_PRIMITIVE_TOPOLOGY   primitive_topology     // tipo di primitiva
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

primitive_topology

[in] Il valore dell'enumerazion ENUM_DX_PRIMITIVE_TOPOLOGY.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

ENUM_DX_PRIMITIVE_TOPOLOGY

ID	Valore	Corrisponde in D3D11_PRIMITIVE_TOPOLOGY
DX_PRIMITIVE_TOPOLOGY_POINTLIST	1	D3D11_PRIMITIVE_TOPOLOGY_POINTLIST
DX_PRIMITIVE_TOPOLOGY_LINELIST	2	D3D11_PRIMITIVE_TOPOLOGY_LINELIST
DX_PRIMITIVE_TOPOLOGY_LINESTRIP	3	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST	4	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP	5	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP
DX_PRIMITIVE_TOPOLOGY_LINELIST_ADJ	6	D3D11_PRIMITIVE_TOPOLOGY_LINELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ	7	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ	8	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ	9	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ

DXBufferSet

Imposta un buffer per il rendering corrente.

```
bool  DXBufferSet(
   int   context,               // handle del contesto grafico
   int   buffer,                // handle del buffer vertex o index
   uint  start=0,               // indice iniziale
   uint  count=WHOLE_ARRAY      // numero di elementi
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

buffer

[in] Handle del buffer vertex o index creato in DXBufferCreate().

start

[in] Indice del primo elemento del buffer. I dati dall'inizio del buffer vengono utilizzati per impostazione predefinita.

count

[in] Numero di valori da utilizzare. L'impostazione predefinita è tutti i valori del buffer.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

Deve essere chiamata la funzione DXBufferSet() per impostare i buffer vertex ed index per il rendering utilizzando DXDraw().

DXShaderSet

Imposta uno shader per il rendering.

```
bool  DXShaderSet(
   int  context,      // handle del contesto grafico
   int  shader        // handle dello shader
   );
```

Parametri

context

[in] Handle per un contesto grafico creato in DXContextCreate().

shader

[in] Handle di uno shader creato in DXShaderCreate().

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

Diversi tipi di shader possono essere utilizzati contemporaneamente per il rendering (vertex, geometry e pixel).

DXHandleType

Restituisce il tipo di handle.

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // handle
   );
```

Parametri

handle

[in] Handle.

Valore di Ritorno

Il valore dall'enumerazione ENUM_DX_HANDLE_TYPE

ENUM_DX_HANDLE_TYPE

ID	Valore	Descrizione
DX_HANDLE_INVALID	0	Handle non valido
DX_HANDLE_CONTEXT	1	Handle del contesto grafico
DX_HANDLE_SHADER	2	Handle shader
DX_HANDLE_BUFFER	3	Handle di vertex buffer o index buffer
DX_HANDLE_INPUT	4	Handle per input shader
DX_HANDLE_TEXTURE	5	Handle texture

DXRelease

Rilascia un handle.

```
bool  DXRelease(
   int  handle      // handle
   );
```

Parametri

context

[in] Handle rilasciato.

Valore di Ritorno

In caso di esecuzione corretta, restituisce true, altrimenti - false. Per ricevere un codice errore, dovrebbe essere chiamata la funzione GetLastError() .

Nota

Tutti gli handle creati che non sono più in uso devono essere esplicitamente rilasciati dalla funzione DXRelease().

Modulo MetaTrader per l'integrazione con Python

MQL5 è progettato per lo sviluppo di applicazioni di trading ad alte prestazioni nei mercati finanziari ed è impareggiabile tra gli altri linguaggi specializzati utilizzati nel trading algoritmico. La sintassi e la velocità dei programmi MQL5 sono il più vicino possibile al C++, c'è un supporto per OpenCL ed integrazione con MS Visual Studio. Statistiche, Logica Fuzzy e ALGLIB sono librerie disponibili, anche. L'ambiente di sviluppo MetaEditor fornisce supporto per le librerie .NET nativo con l'importazione funzioni "intelligente" eliminando la necessità di sviluppare wrapper speciali. È anche possibile utilizzare DLL C++ di terze parti. I file di codice sorgente di C++ (CPP e H) possono essere modificati e compilati in DLL direttamente dall'editor. Microsoft Visual Studio installato sul PC dell'utente può essere utilizzato per questo.

Python è un moderno linguaggio di programmazione ad alto livello per lo sviluppo di script ed applicazioni. Contiene librerie multiple per il machine learning, l'automazione dei processi, nonché l'analisi e la visualizzazione dei dati.

Il pacchetto MetaTrader per Python è progettato per ottenere in modo rapido e conveniente dati di scambio tramite comunicazione interprocessore direttamente dal terminale MetaTrader 5. I dati ricevuti in questo modo possono essere ulteriormente utilizzati per calcoli statistici ed apprendimento automatico.

Installazione del pacchetto dalla riga di comando:

```
pip install MetaTrader5
```

Aggiornamento del pacchetto dalla riga di comando:

```
pip install --upgrade MetaTrader5
```

Funzioni per l'integrazione di MetaTrader 5 e Python

Funzione	Azione
initialize	Stabilire una connessione con il terminale MetaTrader 5
login	Si collega ad un conto di trading utilizzando i parametri specificati
shutdown	Chiude la connessione precedentemente stabilita con il terminale MetaTrader 5
version	Restituisce la versione del terminale MetaTrader 5
last_error	Restituisce i dati sull'ultimo errore
account_info	Ottiene informazioni sull'account di trading corrente
terminal_Info	Ottiene status e parametri del terminale MetaTrader 5 collegato
symbols_total	Ottiene il numero di tutti gli strumenti finanziari nel terminale MetaTrader 5
symbols_get	Ottiene tutti gli strumenti finanziari dal terminale MetaTrader 5
symbol_info	Ottiene dati sullo strumento finanziario specificato
symbol_info_tick	Ottiene l'ultimo tick per lo strumento finanziario specificato

Funzione	Azione
symbol_select	Seleziona un simbolo nella finestra MarketWatch finestra o rimuove un simbolo dalla finestra
market_book_add	Subscribes the MetaTrader 5 terminal to the Market Depth change events for a specified symbol
market_book_get	Returns a tuple from BookInfo featuring Market Depth entries for the specified symbol
market_book_release	Cancels subscription of the MetaTrader 5 terminal to the Market Depth change events for a specified symbol
copy_rates_from	Ottiene barre dal terminale MetaTrader 5 a partire dalla data specificata
copy_rates_from_pos	Ottiene barre dal terminale MetaTrader 5 a partire dall'indice specificato
copyrates_range	Ottiene barre nell'intervallo di date specificato dal terminale MetaTrader 5
copy_ticks_from	Ottiene tick dal terminale MetaTrader 5 a partire dalla data specificata
copy_ticks_range	Ottiene tick per l'intervallo di date specificato dal terminale MetaTrader 5
orders_total	Ottiene il numero di ordini attivi.
orders_get	Ottiene ordini attivi con la possibilità di filtrare per simbolo o ticket
order_calc_margin	Restituisce il margine nella valuta del conto per eseguire un'operazione di trading specificata
order_calc_profit	Restituisce il profitto nella valuta del conto per un'operazione di trading specificata
order_check	Verifica la sufficienza dei fondi per l'esecuzione di un' operazione di trading richiesta
order_send	Invia una richiesta per eseguire un'operazione di trading.
positions_total	Ottiene il numero di posizioni aperte
positions_get	Ottiene le posizioni aperte con la possibilità di filtrare per simbolo o ticket
history_orders_total	Ottiene il numero di ordini nella cronologia di trading entro l'intervallo specificato
history_orders_get	Ottiene ordini dalla cronologia di trading con la possibilità di filtrare per ticket o posizione
history_deals_total	Ottiene il numero di affari nella cronologia di trading entro l'intervallo specificato

Funzione	Azione
history_deals_get	Ottiene affari dalla cronologia di trading con la possibilità di filtrare per biticket o posizione

Esempio di collegamento di Python a MetaTrader 5

1. Scarica l'ultima versione di Python 3.8 dahttps://www.python.org/downloads/windows
2. Quando si installa Python, selezionare "Aggiungi Python 3.8 a PATH%" per poter eseguire gli script Python dalla riga di comando.
3. Installa il modulo MetaTrader 5 dalla riga di comando

```
pip install MetaTrader5
```

4. Aggiungi i pacchetti matplotlib e pandas

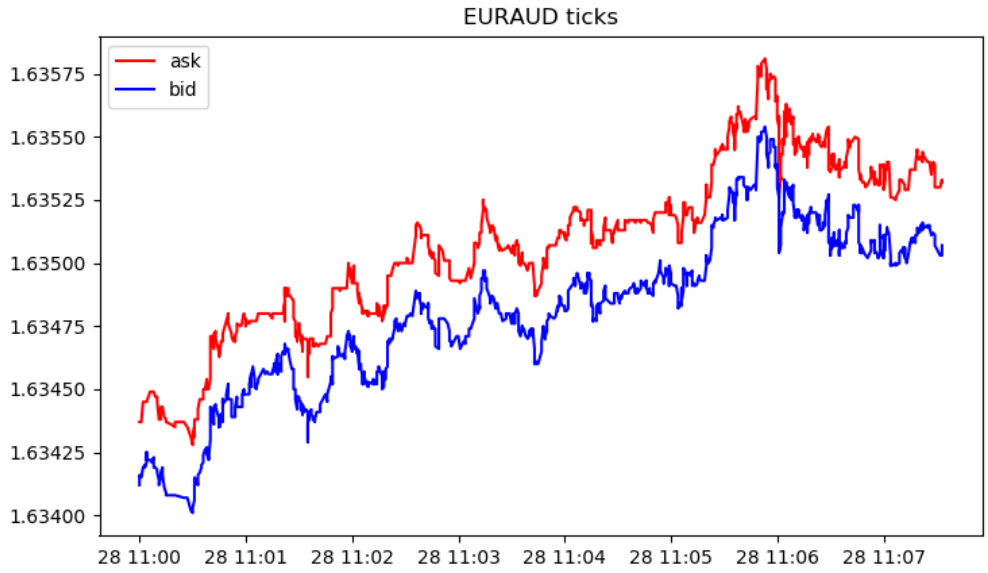
```
pip install matplotlib
pip install pandas
```

5. Avvia lo script di test

```
from datetime import datetime
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import MetaTrader5 as mt5

# connettiti a MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    mt5.shutdown()

# richiesta status e parametri della connessione
print(mt5.terminal_info())
# ottiene dati sulla versione MetaTrader 5
print(mt5.version())

# richiesta 1000 tick da EURAUD
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000, mt5.COPY_T
# richiesta tick da AUDUSD entro il 2019.04.01 13:00 - 2019.04.02 13:00
audusd_ticks = mt5.copy_ticks_range("AUDUSD", datetime(2020,1,27,13), datetime(2020,1,

# ottiene barre da simboli diversi in vari modi
eurusd_rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_M1, datetime(2020,1,28,13),
eurgbp_rates = mt5.copy_rates_from_pos("EURGBP", mt5.TIMEFRAME_M1, 0, 1000)
eurcad_rates = mt5.copy_rates_range("EURCAD", mt5.TIMEFRAME_M1, datetime(2020,1,27,13)

# chiudi la connessione a MetaTrader 5
mt5.shutdown()

#DATA
```

```
print('euraud_ticks(', len(euraud_ticks), ')')
for val in euraud_ticks[:10]: print(val)

print('audusd_ticks(', len(audusd_ticks), ')')
for val in audusd_ticks[:10]: print(val)

print('eurusd_rates(', len(eurusd_rates), ')')
for val in eurusd_rates[:10]: print(val)

print('eurgbp_rates(', len(eurgbp_rates), ')')
for val in eurgbp_rates[:10]: print(val)

print('eurcad_rates(', len(eurcad_rates), ')')
for val in eurcad_rates[:10]: print(val)

#PLOT
# create DataFrame dai dati ottenuti
ticks_frame = pd.DataFrame(euraud_ticks)
# converti il tempo in secondi nel formato datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')
# visualizza ticks sul chart
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# visualizza le legende
plt.legend(loc='upper left')

# aggiungi l'intestazione
plt.title('EURAUD ticks')

# visualizza il chart
plt.show()
```

6. Ottiene dati e chart




```
[2, 'MetaQuotes-Demo', '16167573']
[500, 2325, '19 Feb 2020']

euraud_ticks( 1000 )
(1580209200, 1.63412, 1.63437, 0., 0, 1580209200067, 130, 0.)
(1580209200, 1.63416, 1.63437, 0., 0, 1580209200785, 130, 0.)
(1580209201, 1.63415, 1.63437, 0., 0, 1580209201980, 130, 0.)
(1580209202, 1.63419, 1.63445, 0., 0, 1580209202192, 134, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203004, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203487, 130, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203694, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203990, 130, 0.)
(1580209204, 1.63421, 1.63445, 0., 0, 1580209204194, 130, 0.)
(1580209204, 1.63425, 1.63445, 0., 0, 1580209204392, 130, 0.)
audusd_ticks( 40449 )
(1580122800, 0.67858, 0.67868, 0., 0, 1580122800244, 130, 0.)
(1580122800, 0.67858, 0.67867, 0., 0, 1580122800429, 4, 0.)
(1580122800, 0.67858, 0.67865, 0., 0, 1580122800817, 4, 0.)
(1580122801, 0.67858, 0.67866, 0., 0, 1580122801618, 4, 0.)
(1580122802, 0.67858, 0.67865, 0., 0, 1580122802928, 4, 0.)
(1580122809, 0.67855, 0.67865, 0., 0, 1580122809526, 130, 0.)
(1580122809, 0.67855, 0.67864, 0., 0, 1580122809699, 4, 0.)
(1580122813, 0.67855, 0.67863, 0., 0, 1580122813576, 4, 0.)
(1580122815, 0.67856, 0.67863, 0., 0, 1580122815190, 130, 0.)
(1580122815, 0.67855, 0.67863, 0., 0, 1580122815479, 130, 0.)
eurusd_rates( 1000 )
(1580149260, 1.10132, 1.10151, 1.10131, 1.10149, 44, 1, 0)
(1580149320, 1.10149, 1.10161, 1.10143, 1.10154, 42, 1, 0)
(1580149380, 1.10154, 1.10176, 1.10154, 1.10174, 40, 2, 0)
(1580149440, 1.10174, 1.10189, 1.10168, 1.10187, 47, 1, 0)
```

```
(1580149500, 1.10185, 1.10191, 1.1018, 1.10182, 53, 1, 0)
(1580149560, 1.10182, 1.10184, 1.10176, 1.10183, 25, 3, 0)
(1580149620, 1.10183, 1.10187, 1.10177, 1.10187, 49, 2, 0)
(1580149680, 1.10187, 1.1019, 1.1018, 1.10187, 53, 1, 0)
(1580149740, 1.10187, 1.10202, 1.10187, 1.10198, 28, 2, 0)
(1580149800, 1.10198, 1.10198, 1.10183, 1.10188, 39, 2, 0)
eurgbp_rates( 1000 )
(1582236360, 0.83767, 0.83767, 0.83764, 0.83765, 23, 9, 0)
(1582236420, 0.83765, 0.83765, 0.83764, 0.83765, 15, 8, 0)
(1582236480, 0.83765, 0.83766, 0.83762, 0.83765, 19, 7, 0)
(1582236540, 0.83765, 0.83768, 0.83758, 0.83763, 39, 6, 0)
(1582236600, 0.83763, 0.83768, 0.83763, 0.83767, 21, 6, 0)
(1582236660, 0.83767, 0.83775, 0.83765, 0.83769, 63, 5, 0)
(1582236720, 0.83769, 0.8377, 0.83758, 0.83764, 40, 7, 0)
(1582236780, 0.83766, 0.83769, 0.8376, 0.83766, 37, 6, 0)
(1582236840, 0.83766, 0.83772, 0.83763, 0.83772, 22, 6, 0)
(1582236900, 0.83772, 0.83773, 0.83768, 0.8377, 36, 5, 0)
eurcad_rates( 1441 )
(1580122800, 1.45321, 1.45329, 1.4526, 1.4528, 146, 15, 0)
(1580122860, 1.4528, 1.45315, 1.45274, 1.45301, 93, 15, 0)
(1580122920, 1.453, 1.45304, 1.45264, 1.45264, 82, 15, 0)
(1580122980, 1.45263, 1.45279, 1.45231, 1.45277, 109, 15, 0)
(1580123040, 1.45275, 1.4528, 1.45259, 1.45271, 53, 14, 0)
(1580123100, 1.45273, 1.45285, 1.45269, 1.4528, 62, 16, 0)
(1580123160, 1.4528, 1.45284, 1.45267, 1.45282, 64, 14, 0)
(1580123220, 1.45282, 1.45299, 1.45261, 1.45272, 48, 14, 0)
(1580123280, 1.45272, 1.45275, 1.45255, 1.45275, 74, 14, 0)
(1580123340, 1.45275, 1.4528, 1.4526, 1.4528, 94, 13, 0)
```

initialize

Stabilisce una connessione con il terminale MetaTrader 5. Esistono tre opzioni di chiamata.

Chiamata senza parametri. Il terminale per la connessione viene trovato automaticamente.

```
initialize()
```

Chiamata specificando il percorso del terminale MetaTrader 5 a cui vogliamo collegarci.

```
initialize(
   path                       // percorso al file EXE del terminale MetaTrader 5
   )
```

Chiamata specificando il percorso e i parametri dell'account di trading.

```
initialize(
   path,                      // percorso al file EXE del terminale MetaTrader 5
   login=LOGIN,               // account number
   password="PASSWORD",       // password
   server="SERVER",           // nome del server come specificato nel terminale
   timeout=TIMEOUT,           // timeout
   portable=False             // modalità portatile
   )
```

Parametri

path

[in] Percorso del file metatrader.exe o metatrader64.exe. Parametro unnamed opzionale. Viene indicato per primo senza un nome di parametro. Se il percorso non è specificato, il modulo tenta di trovare il file eseguibile da solo.

login=LOGIN

[in] Numero del conto di trading. Parametro named opzionale. Se non specificato, viene utilizzato l'ultimo conto di trading.

password="PASSWORD"

[in] Password del conto di trading. Parametro named opzionale. Se la password non è impostata, la password per un determinato conto di trading salvato nel database del terminale viene applicata automaticamente.

server="SERVER"

[in] Nome del trade server. Parametro named opzionale. Se il server non è impostato, il server per un account di trading specificato salvato nel database del terminale viene applicato automaticamente.

timeout=TIMEOUT

[in] Timeout della connessione in millisecondi. Parametro named opzionale. Se non specificato, viene applicato il valore di 60.000 (60 secondi).

portable=False

[in] Flag del lancio del terminale in modalità portatile. Parametro named opzionale. Se non specificato, viene utilizzato il valore di False.

Valore di Ritorno

Restituisce True in caso di connessione riuscita al terminale MetaTrader 5, altrimenti - False.

Nota

Se necessario, il terminale MetaTrader 5 viene avviato per stabilire la connessione durante l'esecuzione della chiamata initialize().

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione MetaTrader 5 ad un conto di trading specificato
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# visualizza i dati sullo stato della connessione, il nome del server e il conto di tr
print(mt5.terminal_info())
# visualizza i dati sulla versione MetaTrader 5
print(mt5.version())

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

shutdown, terminal_info, version

login

Collegamento ad un conto di trading utilizzando i parametri specificati.

```
login(
   login,                    // numero di conto
   password="PASSWORD",      // password
   server="SERVER",          // nome del server come specificato nel terminale
   timeout=TIMEOUT           // timeout
   )
```

Parametri

login

[in] Numero del conto di trading. Parametro richiesto senza nome.

password

[in] Password del conto di trading. Parametro named opzionale. Se la password non è impostata, la password salvata nel database del terminale viene applicata automaticamente.

server

[in] Nome del trade server. Parametro named opzionale. Se non è impostato alcun server, l'ultimo server utilizzato viene applicato automaticamente.

timeout=TIMEOUT

[in] Timeout della connessione in millisecondi. Parametro named opzionale. Se non specificato, viene applicato il valore di 60.000 (60 secondi). Se la connessione non viene stabilita entro il tempo specificato, la chiamata viene forzatamente chiusa e viene generata l'eccezione.

Valore di Ritorno

True in caso di connessione riuscita all'account di trade, altrimenti - False.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# visualizza i dati sulla versione MetaTrader 5
print(mt5.version())
# connessione all'account di trade senza specificare una password ed un server
account=17221085
authorized=mt5.login(account)  # la password del database del terminale viene applicat
if authorized:
    print("connesso all'account #{}".format(account))
else:
```

```
    print("fallimento nella connessione all'account #{}, error code: {}".format(accour

# ora si connette ad un altro account di trading specificando la password
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
# visualizza i dati dell'account di trading 'così come sono'
    print(mt5.account_info())
    # visualizza i dati del conto di trading sotto forma di un elenco
    print("Show account_info()._asdict():")
    account_info_dict = mt5.account_info()._asdict()
    for prop in account_info_dict:
        print("  {}={}".format(prop, account_info_dict[prop]))
else:
    print("fallimento nella connessione all'account #{}, error code: {}".format(accour

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()


Result:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
[500, 2367, '23 Mar 2020']

collegato all'account #17221085

collegato all'account #25115284
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
proprietà dell'account:
   login=25115284
   trade_mode=0
   leverage=100
   limit_orders=200
   margin_so_mode=0
   trade_allowed=True
   trade_expert=True
   margin_mode=2
   currency_digits=2
   fifo_close=False
   balance=99588.33
   credit=0.0
   profit=-45.23
   equity=99543.1
   margin=54.37
   margin_free=99488.73
   margin_level=183084.6054809638
   margin_so_call=50.0
   margin_so_so=30.0
```

```
    margin_initial=0.0
    margin_maintenance=0.0
    assets=0.0
    liabilities=0.0
    commission_blocked=0.0
    name=James Smith
    server=MetaQuotes-Demo
    currency=USD
    company=MetaQuotes Software Corp.
```

See also

initialize, shutdown

shutdown

Chiude la connessione precedentemente stabilita con il terminale MetaTrader 5.

```
shutdown()
```

Valore di Ritorno

Nessuna.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    quit()

# visualizza i dati sullo stato della connessione, il nome del server e il conto di t
print(mt5.terminal_info())
# visualizza i dati sulla versione MetaTrader 5
print(mt5.version())

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

initialize, login_py, terminal_info, version

version

Restituisce la versione del terminale MetaTrader 5.

```
version()
```

Valore di Ritorno

Restituisce la versione del terminale MetaTrader 5, la build e la data di rilascio. Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione version() restituisce la versione del terminale, la build e la data di rilascio come una tupla di tre valori:

Tipo	Descrizione	Valore di esempio
integer	Versione del terminale MetaTrader 5	500
integer	Build	2007
string	Data di rilascio della Build	'25 Feb 2019'

Esempio:

```
import MetaTrader5 as mt5
import pandas as pd
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# visualizza i dati sulla versione MetaTrader 5
print(mt5.version())

# visualizza i dati sullo status della connessione, sul nome del server e sull'account
print(mt5.terminal_info())
print()

# ottiene proprietà sotto forma di dizionario
terminal_info_dict=mt5.terminal_info()._asdict()
# converte il dizionario in DataFrame e stampa
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() as dataframe:")
print(df[:-1])
```

```
# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()

Result:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
[500, 2367, '23 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True, dlls_a

terminal_info() as dataframe:
                  property                          value
0       community_account                           True
1    community_connection                           True
2               connected                           True
3            dlls_allowed                          False
4           trade_allowed                          False
5       tradeapi_disabled                          False
6           email_enabled                          False
7             ftp_enabled                          False
8   notifications_enabled                          False
9                    mqid                          False
10                  build                           2367
11                maxbars                           5000
12               codepage                           1251
13             ping_last                          77881
14      community_balance                        707.107
15         retransmission                              0
16                company     MetaQuotes Software Corp.
17                   name                   MetaTrader 5
18               language                        Russian
19                   path  E:\ProgramFiles\MetaTrader 5
20              data_path  E:\ProgramFiles\MetaTrader 5
```

See also

initialize, shutdown, terminal_info

last_error

Restituisce i dati sull'ultimo errore.

```
last_error()
```

Valore di Ritorno

Restituisce l'ultimo codice di errore e la descrizione come tupla.

Nota

last_error() consente di ottenere un codice di errore in caso di mancata esecuzione di una funzione della libreria MetaTrader 5. È simile a GetLastError(). Tuttavia, applica i propri codici di errore. Valori possibili:

Costante	Valore	Descrizione
RES_S_OK	1	successo generico
RES_E_FAIL	-1	errore generico
RES_E_INVALID_PARAMS	-2	argomenti/parametri non validi
RES_E_NO_MEMORY	-3	nessuna condizione di memoria
RES_E_NOT_FOUND	-4	nessuno storico
RES_E_INVALID_VERSION	-5	versione non valida
RES_E_AUTH_FAILED	-6	autorizzazione fallita
RES_E_UNSUPPORTED	-7	metodo non supportato
RES_E_AUTO_TRADING_DISABLED	-8	auto-trading disabilitato
RES_E_INTERNAL_FAIL	-10000	errore generale IPC interno
RES_E_INTERNAL_FAIL_SEND	-10001	send IPC interno non riuscito
RES_E_INTERNAL_FAIL_RECEIVE	-10002	recv interno IPC non riuscito
RES_E_INTERNAL_FAIL_INIT	-10003	inizializzazione IPC interna non riuscita
RES_E_INTERNAL_FAIL_CONNECT	-10003	IPC interno no ipc
RES_E_INTERNAL_FAIL_TIMEOUT	-10005	timeout interno

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
```

```
    quit()

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

version, GetLastError

account_info

Ottiene informazioni sull'account di trading corrente.

```
account_info()
```

Valore di Ritorno

Restituisce informazioni sotto forma di una struttura denominata tupla (namedtuple). Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione restituisce tutti i dati che è possibile ottenere utilizzando AccountInfoInteger, AccountInfoDouble e AccountInfoString in una chiamata.

Esempio:

```
import MetaTrader5 as mt5
import pandas as pd
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# si connette all'account di trade specificando una password e un server
authorized=mt5.login(25115284, password="gqz03431bdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # visualizza i dati dell'account di trading 'così come sono'
        print(account_info)
        # visualizza i dati dell'account di trading sotto forma di dizionario
        print("Show account_info()._asdict():")
        account_info_dict = mt5.account_info()._asdict()
        for prop in account_info_dict:
            print("  {}={}".format(prop, account_info_dict[prop]))
        print()

        # converte il dizionario in DataFrame e stampa
        df=pd.DataFrame(list(account_info_dict.items()),columns=['property','value'])
        print("account_info() as dataframe:")
        print(df)
else:
    print("fallimento nel connettersi al trade account 25115284 con password=gqz03431
```

```
# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()


Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
Show account_info()._asdict():
  login=25115284
  trade_mode=0
  leverage=100
  limit_orders=200
  margin_so_mode=0
  trade_allowed=True
  trade_expert=True
  margin_mode=2
  currency_digits=2
  fifo_close=False
  balance=99511.4
  credit=0.0
  profit=41.82
  equity=99553.22
  margin=98.18
  margin_free=99455.04
  margin_level=101398.67590140559
  margin_so_call=50.0
  margin_so_so=30.0
  margin_initial=0.0
  margin_maintenance=0.0
  assets=0.0
  liabilities=0.0
  commission_blocked=0.0
  server=MetaQuotes-Demo
  currency=USD
  company=MetaQuotes Software Corp.

account_info() as dataframe
              property                     value
0                login                  25115284
1           trade_mode                         0
2             leverage                       100
3         limit_orders                       200
4       margin_so_mode                         0
5        trade_allowed                      True
6         trade_expert                      True
7          margin_mode                         2
8      currency_digits                         2
9           fifo_close                     False
```

```
10             balance                      99588.3
11              credit                            0
12              profit                       -45.13
13              equity                      99543.2
14              margin                        54.37
15         margin_free                      99488.8
16        margin_level                       183085
17      margin_so_call                           50
18        margin_so_so                           30
19      margin_initial                            0
20  margin_maintenance                            0
21              assets                            0
22         liabilities                            0
23  commission_blocked                            0
24                name                   James Smith
25              server               MetaQuotes-Demo
26            currency                          USD
27             company  MetaQuotes Software Corp.
```

See also

initialize, shutdown, login

terminal_info

Ottiene lo status e le impostazioni del terminale client MetaTrader 5 collegato.

```
terminal_info()
```

Valore di Ritorno

Restituisce informazioni sotto forma di una struttura denominata tupla (namedtuple). Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione restituisce tutti i dati che è possibile ottenere utilizzando TerminalInfoInteger, TerminalInfoDouble e TerminalInfoDouble in una chiamata.

Esempio:

```
import MetaTrader5 as mt5
import pandas as pd
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# visualizza i dati sulla versione MetaTrader 5
print(mt5.version())
# visualizza informazioni sulle impostazioni e sullo stato del terminale
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # visualizza i dati del terminale 'come sono'
    print(terminal_info)
    # visualizza i dati sotto forma di un elenco
    print("Show terminal_info()._asdict():")
    terminal_info_dict = mt5.terminal_info()._asdict()
    for prop in terminal_info_dict:
        print("  {}={}".format(prop, terminal_info_dict[prop]))
    print()
    # converte il dizionario in DataFrame e stampa
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() as dataframe:")
    print(df)
```

```
# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()


Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
[500, 2366, '20 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True,....
Show terminal_info()._asdict():
  community_account=True
  community_connection=True
  connected=True
  dlls_allowed=False
  trade_allowed=False
  tradeapi_disabled=False
  email_enabled=False
  ftp_enabled=False
  notifications_enabled=False
  mqid=False
  build=2366
  maxbars=5000
  codepage=1251
  ping_last=77850
  community_balance=707.10668201585
  retransmission=0.0
  company=MetaQuotes Software Corp.
  name=MetaTrader 5
  language=Russian
  path=E:\ProgramFiles\MetaTrader 5
  data_path=E:\ProgramFiles\MetaTrader 5
  commondata_path=C:\Users\Rosh\AppData\Roaming\MetaQuotes\Terminal\Common

terminal_info() as dataframe:
                  property                      value
0       community_account                       True
1    community_connection                       True
2               connected                       True
3            dlls_allowed                      False
4           trade_allowed                      False
5       tradeapi_disabled                      False
6           email_enabled                      False
7             ftp_enabled                      False
8   notifications_enabled                      False
9                    mqid                      False
10                  build                       2367
11                maxbars                       5000
12               codepage                       1251
13             ping_last                      80953
```

```
14      community_balance                    707.107
15         retransmission                   0.063593
16                company  MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                     Russian
```

See also

initialize, shutdown, version

symbols_total

Ottiene il numero di tutti gli strumenti finanziari nel terminale MetaTrader 5.

```
symbols_total()
```

Valore di Ritorno

Valore Integer.

Nota

La funzione è simile a SymbolsTotal(). Tuttavia, restituisce il numero di tutti i simboli inclusi quelli custom e quelli disabilitati in MarketWatch.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene il numero di strumenti finanziari
symbols=mt5.symbols_total()
if symbols>0:
    print("Simboli totali =",symbols)
else:
    print("simboli non trovati")

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

symbols_get, symbol_select, symbol_info

symbols_get

Ottiene tutti gli strumenti finanziari dal terminale MetaTrader 5.

```
symbols_get(
   group="GROUP"      // filtro di selezione del simbolo
)
```

group="GROUP"

[in] Il filtro per disporre un gruppo di simboli necessari. Parametro opzionale. Se viene specificato il gruppo, la funzione restituisce solo simboli che soddisfano un criterio specificato.

Valore di Ritorno

Restituisce i simboli sotto forma di una tupla. Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

Il parametro *group* consente di ordinare i simboli per nome. '*' può essere usato all'inizio e alla fine di una stringa.

Il parametro *group* può essere utilizzato come uno named o unnamed. Entrambe le opzioni funzionano allo stesso modo. L'opzione named (group= "GROUP") semplifica la lettura del codice.

Il parametro *group* può contenere diverse condizioni separate da virgola. Una condizione può essere impostata come maschera usando '*'. Il simbolo di negazione logica "!" può essere utilizzato per un'esclusione. Tutte le condizioni vengono applicate in sequenza, il che significa che le condizioni di inclusione in un gruppo devono essere specificate prima, seguite da una condizione di esclusione. Per esempio, *group*= "*, !EUR" significa che tutti i simboli dovrebbero essere selezionati prima e quelli che contengono "EUR" nei loro nomi dovrebbero essere esclusi in seguito.

Diversamente da symbol_info(), la funzione symbols_get() restituisce i dati su tutti i simboli richiesti all'interno di una singola chiamata.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene tutti i simboli
symbols=mt5.symbols_get()
print('Simboli: ', len(symbols))
count=0
# visualizza i primi cinque
for s in symbols:
```

```
    count+=1
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# ottiene simboli contenenti RU nei loro nomi
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# ottiene simboli i cui nomi non contengono USD, EUR, JPY e GBP
group_symbols=mt5.symbols_get(group="*,!*USD*,!*EUR*,!*JPY*,!*GBP*")
print('len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):', len(group_symbols))
for s in group_symbols:
    print(s.name,":",s)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()

Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
Symbols:  84
1. EURUSD
2. GBPUSD
3. USDCHF
4. USDJPY
5. USDCNH

len(*RU*):  8
EURUSD
USDRUB
USDRUR
EURRUR
EURRUB
FORTS.RUB.M5
EURUSD_T20
EURUSD4

len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):  13
AUDCAD : SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_dea
AUDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDSGD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
```

```
CADMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CHFMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
FORTS.RTS.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FORTS.RUB.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FOREX.CHF.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
```

See also

symbols_total, symbol_select, symbol_info

symbol_info

Ottiene dati sullo strumento finanziario specificato.

```
symbol_info(
   symbol      // nome dello strumento finanziario
)
```

symbol

[in] Nome dello strumento finanziario. Parametro richiesto senza nome.

Valore di Ritorno

Restituisce informazioni sotto forma di una struttura denominata tupla (namedtuple). Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione restituisce tutti i dati che è possibile ottenere utilizzando SymbolInfoInteger, SymbolInfoDouble e SymbolInfoString in una chiamata.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# tenta di abilitare la visualizzazione del simbolo EURJPY in MarketWatch
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("Fallimento nel selezionare EURJPY")
    mt5.shutdown()
    quit()

# visualizza le proprietà del simbolo EURJPY
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
    # mostra i dati del terminale 'così come sono'
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
    # mostra le proprietà del simbolo come lista
    print("Show symbol_info(\"EURJPY\")._asdict():")
    symbol_info_dict = mt5.symbol_info("EURJPY")._asdict()
    for prop in symbol_info_dict:
```

```
        print("  {}={}".format(prop, symbol_info_dict[prop]))

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()


Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURJPY: spread = 17   digits = 3
Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585069682
  digits=3
  spread=17
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
  option_mode=0
  option_right=0
  bid=120.024
  bidhigh=120.506
  bidlow=118.798
  ask=120.041
  askhigh=120.526
  asklow=118.828
```

```
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=0.001
trade_tick_value=0.8977708350166538
trade_tick_value_profit=0.8977708350166538
trade_tick_value_loss=0.8978272580355541
trade_tick_size=0.001
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-0.2
swap_short=-1.2
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
price_greeks_gamma=0.0
price_greeks_vega=0.0
price_greeks_rho=0.0
price_greeks_omega=0.0
price_sensitivity=0.0
basis=
category=
currency_base=EUR
```

```
currency_profit=JPY
currency_margin=EUR
bank=
description=Euro vs Japanese Yen
exchange=
formula=
isin=
name=EURJPY
page=http://www.google.com/finance?q=EURJPY
path=Forex\EURJPY
```

See also

account_info, terminal_info

symbol_info_tick

Ottiene l'ultimo TICK per lo strumento finanziario specificato.

```
symbol_info_tick(
   symbol      // nome dello strumento finanziario
)
```

symbol

[in] Nome dello strumento finanziario. Parametro richiesto senza nome.

Valore di Ritorno

Restituisce informazioni sotto forma di tupla. Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione è simile a SymbolInfoTick.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# tentativo di abilitare la visualizzazione di GBPUSD in MarketWatch
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("Fallimento nel selezionare GBPUSD")
    mt5.shutdown()
    quit()

# mostra l'ultimo tick GBPUSD
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# visualizza i valori dei tick sotto forma di un elenco
print("Mostra symbol_info_tick(\"GBPUSD\")._asdict():")
symbol_info_tick_dict = mt5.symbol_info_tick("GBPUSD")._asdict()
for prop in symbol_info_tick_dict:
    print("  {}={}".format(prop, symbol_info_tick_dict[prop]))

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

```
Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
Tick(time=1585070338, bid=1.17264, ask=1.17279, last=0.0, volume=0, time_msc=158507033
Show symbol_info_tick._asdict():
  time=1585070338
  bid=1.17264
  ask=1.17279
  last=0.0
  volume=0
  time_msc=1585070338728
  flags=2
  volume_real=0.0
```

See also

symbol_info, symbol_info

symbol_select

Seleziona un simbolo nella finestra MarketWatch finestra o rimuove un simbolo dalla finestra.

```
symbol_select(
   symbol,      // nome dello strumento finanziario
   enable=None  // abilita o disabilita
)
```

symbol

[in] Nome dello strumento finanziario. Parametro richiesto senza nome.

enable

[in] Switch. Parametro unnamed opzionale. Se 'false', un simbolo dovrebbe essere rimosso dalla finestra di MarketWatch. Altrimenti, dovrebbe essere selezionato nella finestra di MarketWatch. Non è possibile rimuovere un simbolo se al momento sono presenti charts aperti con questo simbolo o se vi sono posizioni aperte.

Valore di Ritorno

True in caso di successo, altrimenti - False.

Nota

La funzione è simile a SymbolSelect.

Esempio:

```
import MetaTrader5 as mt5
import pandas as pd
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# tentativo di abilitare la visualizzazione di EURCAD in MarketWatch
selected=mt5.symbol_select("EURCAD",True)
if not selected:
    print("Fallimento nel selezionare EURCAD, error code =",mt5.last_error())
else:
    symbol_info=mt5.symbol_info("EURCAD")
    print(symbol_info)
    print("EURCAD: currency_base =",symbol_info.currency_base,"  currency_profit =",sy
    print()

    # ottiene le proprietà dei simboli sotto forma di un dizionario
    print("Show symbol_info()._asdict():")
    symbol_info_dict = symbol_info._asdict()
```

```
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
    print()

    # converte il dizionario in DataFrame e stampa
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() as dataframe:")
    print(df)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()


Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURCAD: currency_base = EUR   currency_profit = CAD   currency_margin = EUR

Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585217595
  digits=5
  spread=39
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
```

```
option_mode=0
option_right=0
bid=1.55192
bidhigh=1.55842
bidlow=1.5419800000000001
ask=1.5523099999999999
askhigh=1.55915
asklow=1.5436299999999998
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=1e-05
trade_tick_value=0.7043642408362214
trade_tick_value_profit=0.7043642408362214
trade_tick_value_loss=0.7044535553770941
trade_tick_size=1e-05
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-1.1
swap_short=-0.9
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
```

```
  price_greeks_gamma=0.0
  price_greeks_vega=0.0
  price_greeks_rho=0.0
  price_greeks_omega=0.0
  price_sensitivity=0.0
  basis=
  category=
  currency_base=EUR
  currency_profit=CAD
  currency_margin=EUR
  bank=
  description=Euro vs Canadian Dollar
  exchange=
  formula=
  isin=
  name=EURCAD
  page=http://www.google.com/finance?q=EURCAD
  path=Forex\EURCAD

symbol_info_dict() as dataframe:
         property                                      value
0          custom                                      False
1      chart_mode                                          0
2          select                                       True
3         visible                                       True
4   session_deals                                          0
..            ...                                        ...
91        formula
92           isin
93           name                                     EURCAD
94           page  http://www.google.com/finance?q=EURCAD
95           path                               Forex\EURCAD

[96 rows x 2 columns]
```

See also

symbol_info

market_book_add

Subscribes the MetaTrader 5 terminal to the Market Depth change events for a specified symbol.

```
market_book_add(
   symbol      // financial instrument name
)
```

symbol

[in] Financial instrument name. Required unnamed parameter.

Return Value

True if successful, otherwise - False.

Note

The function is similar to MarketBookAdd.

See also

market_book_get, market_book_release, Market Depth structure

market_book_get

Returns a tuple from BookInfo featuring Market Depth entries for the specified symbol.

```
market_book_get(
   symbol      // financial instrument name
)
```

symbol

[in] Financial instrument name. Required unnamed parameter.

Return Value

Returns the Market Depth content as a tuple from BookInfo entries featuring order type, price and volume in lots. BookInfo is similar to the MqlBookInfo structure.

Return None in case of an error. The info on the error can be obtained using last_error().

Note

The subscription to the Market Depth change events should be preliminarily performed using the market_book_add() function.

The function is similar to MarketBookGet.

Example:

```
import MetaTrader5 as mt5
import time
# display data on the MetaTrader 5 package
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print("")

# establish connection to the MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
   # shut down connection to the MetaTrader 5 terminal
    mt5.shutdown()
    quit()

# subscribe to market depth updates for EURUSD (Depth of Market)
if mt5.market_book_add('EURUSD'):
  # get the market depth data 10 times in a loop
   for i in range(10):
        # get the market depth content (Depth of Market)
        items = mt5.market_book_get('EURUSD')
        # display the entire market depth 'as is' in a single string
        print(items)
        # now display each order separately for more clarity
        if items:
```

```
            for it in items:
                # order content
                print(it._asdict())
        # pause for 5 seconds before the next request of the market depth data
        time.sleep(5)
  # cancel the subscription to the market depth updates (Depth of Market)
   mt5.market_book_release('EURUSD')
else:
    print("mt5.market_book_add('EURUSD') failed, error code =",mt5.last_error())

# shut down connection to the MetaTrader 5 terminal
mt5.shutdown()


Result:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.34

(BookInfo(type=1, price=1.20038, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20038, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20032, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20023, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20017, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20036, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20036, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20028, 'volume': 50, 'volume_dbl': 50.0}
```

```
{'type': 1, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20035, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20035, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20027, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20037, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20037, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20031, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20022, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20016, 'volume': 250, 'volume_dbl': 250.0}
```

See also

market_book_add, market_book_release, Market Depth structure

market_book_release

Cancels subscription of the MetaTrader 5 terminal to the Market Depth change events for a specified symbol.

```
market_book_release(
   symbol      // financial instrument name
)
```

symbol

[in] Financial instrument name. Required unnamed parameter.

Return Value

True if successful, otherwise - False.

Note

The function is similar to MarketBookRelease.

See also

market_book_add, market_book_get, Market Depth structure

copy_rates_from

Ottiene barre dal terminale MetaTrader 5 a partire dalla data specificata.

```
copy_rates_from(
   symbol,       // nome del simbolo
   timeframe,    // timeframe
   date_from,    // data iniziale di apertura della barra
   count         // numero di barre
   )
```

Parametri

symbol

[in] Nome dello strumento finanziario, ad esempio "EURUSD". Parametro richiesto senza nome.

timeframe

[in] Timeframe per cui sono richieste le barre. Impostato da un valore dell'enumerazione TIMEFRAME. Parametro unnamed richiesto.

date_from

[in] Data di apertura della prima barra dal campione richiesto. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

count

[in] Numero di barre da ricevere. Parametro richiesto senza nome.

Valore di Ritorno

Restituisce le barre come matrice numpy con le colonne denominate time, open, high, low, close, tick_volume, spread e real_volume. Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

Vedere la funzione CopyRates() per ulteriori informazioni.

Solo i dati la cui data è minore (prima) o uguale alla data specificata. Vuol dire, che il tempo di apertura di ogni barra è sempre minore o uguale a quello specificato.

Il terminale MetaTrader 5 fornisce barre solo all'interno di una cronologia disponibile per un utente su charts. Il numero di barre disponibili per gli utenti sono impostate nel parametro "Max. barre nel chart".

Durante la creazione dell'oggetto 'datetime', Python utilizza il fuso orario locale, mentre MetaTrader 5 memorizza il tick e la barra dell'ora aperta nel fuso orario UTC (senza lo slittamento). Pertanto, 'datetime' deve essere creato nell'ora UTC per l'esecuzione di funzioni che utilizzano l'ora. I dati ricevuti dal terminale MetaTrader 5 hanno l'ora UTC.

TIMEFRAME è un'enumerazione con i valori del perioro possibili per il chart

ID	Descrizione
TIMEFRAME_M1	1 minuto

ID	Descrizione
TIMEFRAME_M2	2 minuti
TIMEFRAME_M3	3 minuti
TIMEFRAME_M4	4 minuti
TIMEFRAME_M5	5 minuti
TIMEFRAME_M6	6 minuti
TIMEFRAME_M10	10 minuti
TIMEFRAME_M12	12 minuti
TIMEFRAME_M12	15 minuti
TIMEFRAME_M20	20 minuti
TIMEFRAME_M30	30 minuti
TIMEFRAME_H1	1 ora
TIMEFRAME_H2	2 ore
TIMEFRAME_H3	3 ore
TIMEFRAME_H4	4 ore
TIMEFRAME_H6	6 ore
TIMEFRAME_H8	8 ore
TIMEFRAME_H12	12 ore
TIMEFRAME_D1	1 giorno
TIMEFRAME_W1	1 settimana
TIMEFRAME_MN1	1 mese

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importa il modulo 'pandas' per visualizzare i dati ottenuti in forma tabellare
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # max table width to display
# importa il modulo pytz per lavorare con il fuso orario
import pytz
```

```
# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# imposta il fuso orario su UTC
timezone = pytz.timezone("Etc/UTC")
# crea un oggetto 'datetime' nel fuso orario UTC per evitare l'implementazione di un f
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# ottiene 10 barre EUR4 USD H4 a partire dal 01.10.2020 nel fuso orario UTC
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
# visualizza ogni elemento dei dati ottenuti in una nuova riga
print("Mostra i dati ottenuti 'così come sono' '")
for rate in rates:
    print(rate)

# create DataFrame dai dati ottenuti
rates_frame = pd.DataFrame(rates)
# converti il tempo in secondi nel formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# visualizza dati
print("\nMostra dataframe con dati")
print(rates_frame)


Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

Visualizza i dati ottenuti "così come sono"
(1578484800, 1.11382, 1.11385, 1.1111, 1.11199, 9354, 1, 0)
(1578499200, 1.11199, 1.11308, 1.11086, 1.11179, 10641, 1, 0)
(1578513600, 1.11178, 1.11178, 1.11016, 1.11053, 4806, 1, 0)
(1578528000, 1.11053, 1.11193, 1.11033, 1.11173, 3480, 1, 0)
(1578542400, 1.11173, 1.11189, 1.11126, 1.11182, 2236, 1, 0)
(1578556800, 1.11181, 1.11203, 1.10983, 1.10993, 7984, 1, 0)
(1578571200, 1.10994, 1.11173, 1.10965, 1.11148, 7406, 1, 0)
(1578585600, 1.11149, 1.11149, 1.10923, 1.11046, 7468, 1, 0)
(1578600000, 1.11046, 1.11097, 1.11033, 1.11051, 3450, 1, 0)
(1578614400, 1.11051, 1.11093, 1.11017, 1.11041, 2448, 1, 0)

Visualizza dataframe con i dati
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-08 12:00:00  1.11382  1.11385  1.11110  1.11199         9354       1
```

```
1 2020-01-08 16:00:00  1.11199  1.11308  1.11086  1.11179        10641       1
2 2020-01-08 20:00:00  1.11178  1.11178  1.11016  1.11053         4806       1
3 2020-01-09 00:00:00  1.11053  1.11193  1.11033  1.11173         3480       1
4 2020-01-09 04:00:00  1.11173  1.11189  1.11126  1.11182         2236       1
5 2020-01-09 08:00:00  1.11181  1.11203  1.10983  1.10993         7984       1
6 2020-01-09 12:00:00  1.10994  1.11173  1.10965  1.11148         7406       1
7 2020-01-09 16:00:00  1.11149  1.11149  1.10923  1.11046         7468       1
8 2020-01-09 20:00:00  1.11046  1.11097  1.11033  1.11051         3450       1
9 2020-01-10 00:00:00  1.11051  1.11093  1.11017  1.11041         2448       1
```

See also

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_rates_from_pos

Ottiene barre dal terminale MetaTrader 5 a partire dall'indice specificato.

```
copy_rates_from_pos(
   symbol,       // nome del simbolo
   timeframe,    // timeframe
   start_pos,    // indice della barra iniziale
   count         // numero di barre
   )
```

Parametri

symbol

[in] Nome dello strumento finanziario, ad esempio "EURUSD". Parametro richiesto senza nome.

timeframe

[in] Timeframe per cui sono richieste le barre. Impostato da un valore dell'enumerazione TIMEFRAME. Parametro unnamed richiesto.

start_pos

[in] Indice iniziale della barra da cui vengono richiesti i dati. La numerazione delle barre va dal presente al passato. Pertanto, la barra zero indica quella corrente. Parametro richiesto senza nome.

count

[in] Numero di barre da ricevere. Parametro richiesto senza nome.

Valore di Ritorno

Restituisce le barre come matrice numpy con le colonne denominate time, open, high, low, close, tick_volume, spread e real_volume. Restituisce Nessuno in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

Vedere la funzione CopyRates() per ulteriori informazioni.

Il terminale MetaTrader 5 fornisce barre solo all'interno di una cronologia disponibile per un utente su charts. Il numero di barre disponibili per gli utenti sono impostate nel parametro "Max. barre nel chart".

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importa il modulo 'pandas' per visualizzare i dati ottenuti in forma tabellare
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
```

```
pd.set_option('display.width', 1500)      # max table width to display

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene 10 barre D1 GBPUSD dal giorno corrente
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
# visualizza ogni elemento dei dati ottenuti in una nuova riga
print("Mostra i dati ottenuti 'così come sono' '")
for rate in rates:
    print(rate)

# create DataFrame dai dati ottenuti
rates_frame = pd.DataFrame(rates)
# converti il tempo in secondi nel formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# visualizza dati
print("\nMostra dataframe con dati")
print(rates_frame)

Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

Visualizza i dati ottenuti "così come sono"
(1581552000, 1.29568, 1.30692, 1.29441, 1.30412, 68228, 0, 0)
(1581638400, 1.30385, 1.30631, 1.3001, 1.30471, 56498, 0, 0)
(1581897600, 1.30324, 1.30536, 1.29975, 1.30039, 49400, 0, 0)
(1581984000, 1.30039, 1.30486, 1.29705, 1.29952, 62288, 0, 0)
(1582070400, 1.29952, 1.3023, 1.29075, 1.29187, 57909, 0, 0)
(1582156800, 1.29186, 1.29281, 1.28489, 1.28792, 61033, 0, 0)
(1582243200, 1.28802, 1.29805, 1.28746, 1.29566, 66386, 0, 0)
(1582502400, 1.29426, 1.29547, 1.28865, 1.29283, 66933, 0, 0)
(1582588800, 1.2929, 1.30178, 1.29142, 1.30037, 80121, 0, 0)
(1582675200, 1.30036, 1.30078, 1.29136, 1.29374, 49286, 0, 0)

Visualizza dataframe con i dati
        time     open     high      low    close  tick_volume  spread  real_volume
0 2020-02-13  1.29568  1.30692  1.29441  1.30412        68228       0            0
1 2020-02-14  1.30385  1.30631  1.30010  1.30471        56498       0            0
2 2020-02-17  1.30324  1.30536  1.29975  1.30039        49400       0            0
3 2020-02-18  1.30039  1.30486  1.29705  1.29952        62288       0            0
4 2020-02-19  1.29952  1.30230  1.29075  1.29187        57909       0            0
```

5	2020-02-20	1.29186	1.29281	1.28489	1.28792	61033	0	0
6	2020-02-21	1.28802	1.29805	1.28746	1.29566	66386	0	0
7	2020-02-24	1.29426	1.29547	1.28865	1.29283	66933	0	0
8	2020-02-25	1.29290	1.30178	1.29142	1.30037	80121	0	0
9	2020-02-26	1.30036	1.30078	1.29136	1.29374	49286	0	0

See also

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_rates_range

Ottiene barre nell'intervallo di date specificato dal terminale MetaTrader 5.

```
copy_rates_range(
   symbol,       // nome del simbolo
   timeframe,    // timeframe
   date_from,    // data in cui sono richieste le barre
   date_to       // data, fino alla quale sono richieste le barre
   )
```

Parametri

symbol

[in] Nome dello strumento finanziario, ad esempio "EURUSD". Parametro richiesto senza nome.

timeframe

[in] Timeframe per cui sono richieste le barre. Impostato da un valore dell'enumerazione TIMEFRAME. Parametro unnamed richiesto.

date_from

[in] Data di richiesta delle barre. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Vengono restituite le barre con l'orario di apertura >= date_from. Parametro richiesto senza nome.

date_to

[in] Data, fino alla quale sono richieste le barre. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Vengono restituite le barre con l'orario di apertura <= date_to. Parametro richiesto senza nome.

Valore di Ritorno

Restituisce le barre come matrice numpy con le colonne denominate time, open, high, low, close, tick_volume, spread e real_volume. Restituisce Nessuno in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

Vedere la funzione CopyRates() per ulteriori informazioni.

Il terminale MetaTrader 5 fornisce barre solo all'interno di una cronologia disponibile per un utente su charts. Il numero di barre disponibili per gli utenti sono impostate nel parametro "Max. barre nel chart".

Durante la creazione dell'oggetto 'datetime', Python utilizza il fuso orario locale, mentre MetaTrader 5 memorizza il tick e la barra dell'ora aperta nel fuso orario UTC (senza lo slittamento). Pertanto, 'datetime' deve essere creato nell'ora UTC per l'esecuzione di funzioni che utilizzano l'ora. I dati ricevuti dal terminale MetaTrader 5 hanno l'ora UTC.

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
```

```
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importa il modulo 'pandas' per visualzzare i dati ottenuti in forma tabellare
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # max table width to display
# import pytz module for working with time zone
import pytz

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# imposta fuso orario UTC
timezone = pytz.timezone("Etc/UTC")
# crea oggetti 'datetime' nel fuso orario UTC per evitare l'implementazione di un offs
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# ottiene barre da USDJPY M5 nell'intervallo 2020.01.10 00:00 - 2020.01.11 13:00 in UT
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# chiude la connessione a MetaTrader 5 terminal
mt5.shutdown()

# visualizza ogni elemento dei dati ottenuti in una nuova riga
print("Visualizza i dati ottenuti 'così come sono' ")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# crea DataFrame dai dati ottenuti
rates_frame = pd.DataFrame(rates)
# converte il tempo in secondi nel formato 'datetime'
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# display data
print("\nMostra dataframe con i dati")
print(rates_frame.head(10))
```

Risultato:

```
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

Visualizza i dati ottenuti "così come sono"
(1578614400, 109.513, 109.527, 109.505, 109.521, 43, 2, 0)
(1578614700, 109.521, 109.549, 109.518, 109.543, 215, 8, 0)
(1578615000, 109.543, 109.543, 109.466, 109.505, 98, 10, 0)
(1578615300, 109.504, 109.534, 109.502, 109.517, 155, 8, 0)
(1578615600, 109.517, 109.539, 109.513, 109.527, 71, 4, 0)
(1578615900, 109.526, 109.537, 109.484, 109.52, 106, 9, 0)
(1578616200, 109.52, 109.524, 109.508, 109.51, 205, 7, 0)
```

```
(1578616500, 109.51, 109.51, 109.491, 109.496, 44, 8, 0)
(1578616800, 109.496, 109.509, 109.487, 109.5, 85, 5, 0)
(1578617100, 109.5, 109.504, 109.487, 109.489, 82, 7, 0)

Visualizza dataframe con i dati
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-10 00:00:00  109.513  109.527  109.505  109.521           43       2
1 2020-01-10 00:05:00  109.521  109.549  109.518  109.543          215       8
2 2020-01-10 00:10:00  109.543  109.543  109.466  109.505           98      10
3 2020-01-10 00:15:00  109.504  109.534  109.502  109.517          155       8
4 2020-01-10 00:20:00  109.517  109.539  109.513  109.527           71       4
5 2020-01-10 00:25:00  109.526  109.537  109.484  109.520          106       9
6 2020-01-10 00:30:00  109.520  109.524  109.508  109.510          205       7
7 2020-01-10 00:35:00  109.510  109.510  109.491  109.496           44       8
8 2020-01-10 00:40:00  109.496  109.509  109.487  109.500           85       5
9 2020-01-10 00:45:00  109.500  109.504  109.487  109.489           82       7
```

See also

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_ticks_from

Ottiene tick dal terminale MetaTrader 5 a partire dalla data specificata.

```
copy_ticks_from(
   symbol,       // nome del simbolo
   date_from,    // data in cui sono richiesti i tick
   count,        // numero di tick richiesti
   flags         // combinazione di flag che definisce il tipo di tick richiesti
   )
```

Parametri

symbol

[in] Nome dello strumento finanziario, ad esempio "EURUSD". Parametro richiesto senza nome.

from

[in] Data da cui vengono richiesti i tick. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

count

[in] Numero di tick da ricevere. Parametro richiesto senza nome.

flags

[in] Un flag per definire il tipo di tick richiesti. COPY_TICKS_INFO - tick con cambiamenti Bid e/o Ask, COPY_TICKS_TRADE - tick con cambiamenti in Lase e Volume, COPY_TICKS_ALL - tutti i tick. I valori dei Flag sono descritti nell'enumerazione COPY_TICKS. Parametro unnamed richiesto.

Valore di Ritorno

Restituisce tick come l'array numpy con le colonne denominate time, bid, ask, last e flags. Il valore 'flags' può essere una combinazione di flag dall'enumerzione TICK_FLAG. Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

Vedi la funzione CopyTicks per ulteriori informazioni.

Durante la creazione dell'oggetto 'datetime', Python utilizza il fuso orario locale, mentre MetaTrader 5 memorizza il tick e la barra dell'ora aperta nel fuso orario UTC (senza lo slittamento). Pertanto, 'datetime' deve essere creato nell'ora UTC per l'esecuzione di funzioni che utilizzano l'ora. I dati ricevuti dal terminale MetaTrader 5 hanno l'ora UTC.

COPY_TICKS definisce i tipi di tick che possono essere richiesti usando le funzioni copy_ticks_from() e copy_ticks_range().

ID	Descrizione
COPY_TICKS_ALL	tutti i ticks
COPY_TICKS_INFO	ticks contenenti variazioni di prezzi Bid e/o Ask
COPY_TICKS_TRADE	tick contenenti le variazioni del prezzo di Last e/o Volume

TICK_FLAG definisce possibili flag per i ticks. Questi flag sono usati per descrivere i tick ottenuti dalle funzioni copy_ticks_from() e copy_ticks_range().

ID	Descrizione
TICK_FLAG_BID	Il prezzo di Bid è cambiato
TICK_FLAG_ASK	Il prezzo di Ask è cambiato
TICK_FLAG_LAST	Il prezzo Last è cambiato
TICK_FLAG_VOLUME	Il Volume è cambiato
TICK_FLAG_BUY	il prezzo dell'ultimo Buy è cambiato
TICK_FLAG_SELL	il prezzo dell'ultimo Sell è cambiato

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importa il modulo 'pandas' per visualizzare i dati ottenuti in forma tabellare
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # max table width to display
# importa il modulo pytz per lavorare con il fuso orario
import pytz

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# imposta il fuso orario su UTC
timezone = pytz.timezone("Etc/UTC")
# crea un oggetto 'datetime' nel fuso orario UTC per evitare l'implementazione di un
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# richiesta 100.000 tick EURUSD a partire dal 10.01.2019 nel fuso orario UTC
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Tick ricevuti:",len(ticks))

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()

# visualizza i dati su ogni tick su una nuova riga
print("Visualizza i tick ottenuti 'così come sono' ")
count = 0
for tick in ticks:
```

```
    count+=1
    print(tick)
    if count >= 10:
        break

# create DataFrame dai dati ottenuti
ticks_frame = pd.DataFrame(ticks)
# converti il tempo in secondi nel formato datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# display data
print("\nMostra dataframe con i tick")
print(ticks_frame.head(10))

Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

Tick ricevuti: 100000
Visualizza i tick ottenuti 'così come sono'
(1578614400, 1.11051, 1.11069, 0., 0, 1578614400987, 134, 0.)
(1578614402, 1.11049, 1.11067, 0., 0, 1578614402025, 134, 0.)
(1578614404, 1.1105, 1.11066, 0., 0, 1578614404057, 134, 0.)
(1578614404, 1.11049, 1.11067, 0., 0, 1578614404344, 134, 0.)
(1578614412, 1.11052, 1.11064, 0., 0, 1578614412106, 134, 0.)
(1578614418, 1.11039, 1.11051, 0., 0, 1578614418265, 134, 0.)
(1578614418, 1.1104, 1.1105, 0., 0, 1578614418905, 134, 0.)
(1578614419, 1.11039, 1.11051, 0., 0, 1578614419519, 134, 0.)
(1578614456, 1.11037, 1.11065, 0., 0, 1578614456011, 134, 0.)
(1578614456, 1.11039, 1.11051, 0., 0, 1578614456015, 134, 0.)

Visualizza il dataframe con ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  1.11051  1.11069   0.0       0  1578614400987    134          0
1 2020-01-10 00:00:02  1.11049  1.11067   0.0       0  1578614402025    134          0
2 2020-01-10 00:00:04  1.11050  1.11066   0.0       0  1578614404057    134          0
3 2020-01-10 00:00:04  1.11049  1.11067   0.0       0  1578614404344    134          0
4 2020-01-10 00:00:12  1.11052  1.11064   0.0       0  1578614412106    134          0
5 2020-01-10 00:00:18  1.11039  1.11051   0.0       0  1578614418265    134          0
6 2020-01-10 00:00:18  1.11040  1.11050   0.0       0  1578614418905    134          0
7 2020-01-10 00:00:19  1.11039  1.11051   0.0       0  1578614419519    134          0
8 2020-01-10 00:00:56  1.11037  1.11065   0.0       0  1578614456011    134          0
9 2020-01-10 00:00:56  1.11039  1.11051   0.0       0  1578614456015    134          0
```

See also

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_ticks_range

Ottiene tick per l'intervallo di date specificato dal terminale MetaTrader 5.

```
copy_ticks_range(
   symbol,       // nome del simbolo
   date_from,    // data in cui sono richiesti i tick
   date_to,      // data, fino alla quale sono richieste le
   flags         // combinazione di flag che definisce il tipo di tick richiesti
   )
```

Parametri

symbol

[in] Nome dello strumento finanziario, ad esempio "EURUSD". Parametro richiesto senza nome.

date_from

[in] Data da cui vengono richiesti i tick. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

date_to

[in] Data, fino alla quale sono richiesti i tick. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

flags

[in] Un flag per definire il tipo di tick richiesti. COPY_TICKS_INFO - tick con cambiamenti Bid e/o Ask, COPY_TICKS_TRADE - tick con cambiamenti in Lase e Volume, COPY_TICKS_ALL - tutti i tick. I valori dei Flag sono descritti nell'enumerazione COPY_TICKS. Parametro unnamed richiesto.

Valore di Ritorno

Restituisce tick come l'array numpy con le colonne denominate time, bid, ask, last e flags. Il valore 'flags' può essere una combinazione di flag dall'enumerzione TICK_FLAG. Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

Vedi la funzione CopyTicks per ulteriori informazioni.

Durante la creazione dell'oggetto 'datetime', Python utilizza il fuso orario locale, mentre MetaTrader 5 memorizza il tick e la barra dell'ora aperta nel fuso orario UTC (senza lo slittamento). Pertanto, 'datetime' deve essere creato nell'ora UTC per l'esecuzione di funzioni che utilizzano l'ora. I dati ottenuti da MetaTrader 5 hanno l'ora UTC, ma Python applica nuovamente il fuso orario locale quando tenta di stamparli. Pertanto, anche i dati ottenuti devono essere corretti per la presentazione visiva.

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
```

```
# importa il modulo 'pandas' per visualzzare i dati ottenuti in forma tabellare
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # max table width to display
# import pytz module for working with time zone
import pytz

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# imposta il fuso orario su UTC
timezone = pytz.timezone("Etc/UTC")
# crea oggetti 'datetime' nel fuso orario UTC per evitare l'implementazione di un fuso
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# richiesta tick di AUDUSD entro 11.01.2020 - 11.01.2020
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Tick ricevuti:",len(ticks))

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()

# visualizza i dati su ogni tick su una nuova riga
print("Visualizza i ticks ottenuti 'così come sono' ")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# crea DataFrame dai dati ottenuti
ticks_frame = pd.DataFrame(ticks)
# converti il tempo in secondi nel formato datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# display data
print("\nDisplay dataframe with ticks")
print(ticks_frame.head(10))
```

```
Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

Tick ricevuti: 37008
Visualizza i tick ottenuti 'così come sono'
(1578614400, 0.68577, 0.68594, 0., 0, 1578614400820, 134, 0.)
(1578614401, 0.68578, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614401, 0.68575, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614411, 0.68576, 0.68594, 0., 0, 1578614411388, 130, 0.)
(1578614411, 0.68575, 0.68594, 0., 0, 1578614411560, 130, 0.)
(1578614414, 0.68576, 0.68595, 0., 0, 1578614414973, 134, 0.)
(1578614430, 0.68576, 0.68594, 0., 0, 1578614430188, 4, 0.)
(1578614450, 0.68576, 0.68595, 0., 0, 1578614450408, 4, 0.)
```

```
(1578614450, 0.68576, 0.68594, 0., 0, 1578614450519, 4, 0.)
(1578614456, 0.68575, 0.68594, 0., 0, 1578614456363, 130, 0.)

Visualizza il dataframe con ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  0.68577  0.68594   0.0       0  1578614400820    134          0
1 2020-01-10 00:00:01  0.68578  0.68594   0.0       0  1578614401128    130          0
2 2020-01-10 00:00:01  0.68575  0.68594   0.0       0  1578614401128    130          0
3 2020-01-10 00:00:11  0.68576  0.68594   0.0       0  1578614411388    130          0
4 2020-01-10 00:00:11  0.68575  0.68594   0.0       0  1578614411560    130          0
5 2020-01-10 00:00:14  0.68576  0.68595   0.0       0  1578614414973    134          0
6 2020-01-10 00:00:30  0.68576  0.68594   0.0       0  1578614430188      4          0
7 2020-01-10 00:00:50  0.68576  0.68595   0.0       0  1578614450408      4          0
8 2020-01-10 00:00:50  0.68576  0.68594   0.0       0  1578614450519      4          0
9 2020-01-10 00:00:56  0.68575  0.68594   0.0       0  1578614456363    130          0
```

See also

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

orders_total

Ottiene il numero di ordini attivi.

```
orders_total()
```

Valore di Ritorno

Valore Integer.

Nota

La funzione è simile a OrdersTotal.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# verifica la presenza di ordini attivi
orders=mt5.orders_total()
if orders>0:
    print("Totale ordini=",orders)
else:
    print("Ordini non trovati")

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

orders_get, positions_total

orders_get

Ottiene ordini attivi con la possibilità di filtrare per simbolo o ticket. Esistono tre opzioni di chiamata.

Chiamata senza parametri. Restituisce gli ordini attivi su tutti i simboli.

```
orders_get()
```

Chiama specificando un simbolo per cui devono essere ricevuti gli ordini attivi.

```
orders_get(
   symbol="SYMBOL"      // nome simbolo
)
```

Chiama specificando un gruppo di simboli per cui devono essere ricevuti gli ordini attivi.

```
orders_get(
   group="GROUP"        // filtro per selezionare gli ordini, per simboli
)
```

Chiama specificando il ticket dell'ordine.

```
orders_get(
   ticket=TICKET        // ticket
)
```

symbol="SYMBOL"

[in] Nome simbolo. Parametro named opzionale. Se viene specificato un simbolo, il parametro *ticket* viene ignorato.

group="GROUP"

[in] Il filtro per disporre un gruppo di simboli necessari. Parametro named opzionale. Se viene specificato il gruppo, la funzione restituisce solo gli ordini attivi che soddisfano un criterio specificato per un nome simbolo.

ticket=TICKET

[in] Order ticket (ORDER_TICKET). Parametro named facoltativo.

Valore di Ritorno

Restituisce informazioni sotto forma di una struttura denominata tupla (namedtuple). Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione consente di ricevere tutti gli ordini attivi all'interno di una chiamata similmente al tandem OrdersTotal e OrderSelect.

Il parametro *group* consente di ordinare gli ordini in base ai simboli. '*' può essere usato all'inizio e alla fine di una stringa.

Il parametro *group* può contenere diverse condizioni separate da virgola. Una condizione può essere impostata come maschera usando '*'. Il simbolo di negazione logica "!" può essere utilizzato per un'esclusione. Tutte le condizioni vengono applicate in sequenza, il che significa che le condizioni di inclusione in un gruppo devono essere specificate prima, seguite da una condizione di esclusione.

Per esempio, *group*= "*, !EUR" significa che gli ordini per tutti i simboli devono essere selezionati prima, e quelli che contengono "EUR" nei nomi dei simboli devono essere esclusi in seguito.

Esempio:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # larghezza massima della tabella da visuali
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# visualizza i dati sugli ordini attivi su GBPUSD
orders=mt5.orders_get(symbol="GBPUSD")
if orders==None:
    print("Niente ordini su GBPUSD, error code={}".format(mt5.last_error()))
elif len(orders)>0:
    print("Totale ordini su GBPUSD:",len(orders))
    # mostra tutti gli ordini attivi
    for order in orders:
        print(order)
print()

# ottiene l'elenco degli ordini sui simboli i cui nomi contengono "*GBP*"
gbp_orders=mt5.orders_get(group="*GBP*")
if gbp_orders==None:
    print("Niente ordini col gruppo=\"*GBP*\", error code={}".format(mt5.last_error()))
elif len(gbp_orders)>0:
    print("orders_get(group=\"*GBP*\")={}".format(len(gbp_orders)))
    # visualizza questi ordini come una tabella usando pandas.DataFrame
    df=pd.DataFrame(list(gbp_orders),columns=gbp_orders[0]._asdict().keys())
    df.drop(['time_done', 'time_done_msc', 'position_id', 'position_by_id', 'reason',
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    print(df)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()

Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

Totale ordini su GBPUSD: 2
```

```
TradeOrder(ticket=554733548, time_setup=1585153667, time_setup_msc=1585153667718, time
TradeOrder(ticket=554733621, time_setup=1585153671, time_setup_msc=1585153671419, time

orders_get(group="*GBP*")=4
      ticket           time_setup  time_setup_msc  time_expiration  type  type_time  ty
0  554733548 2020-03-25 16:27:47   1585153667718                0     3          0
1  554733621 2020-03-25 16:27:51   1585153671419                0     2          0
2  554746664 2020-03-25 16:38:14   1585154294401                0     3          0
3  554746710 2020-03-25 16:38:17   1585154297022                0     2          0
```

See also

orders_total, positions_get

order_calc_margin

Restituisce il margine nella valuta del conto per eseguire un'operazione di trading specificata.

```
order_calc_margin(
   action,      // tipo di ordine (ORDER_TYPE_BUY or ORDER_TYPE_SELL)
   symbol,      // nome del simbolo
   volume,      // volume
   price        // prezzo di apertura
   )
```

Parametri

action

[in] Tipo di ordine che prende valori dall'enumerazione ORDER_TYPE. Parametro unnamed richiesto.

symbol

[in] Nome dello strumento finanziario. Parametro richiesto senza nome.

volume

[in] Volume delle operazioni di trading. Parametro richiesto senza nome.

price

[in] Prezzo di apertura. Parametro richiesto senza nome.

Valore di Ritorno

Valore Real in caso di successo, altrimenti Nessuno. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione consente di stimare il margine necessario per un determinato tipo di ordine sul conto attuale e nell'attuale contesto di mercato senza considerare le posizioni aperte e gli ordini in sospeso. La funzione è simile a OrderCalcMargin.

ORDER_TYPE

ID	Descrizione
ORDER_TYPE_BUY	Ordine di Buy sul mercato
ORDER_TYPE_SELL	Ordine di Sell sul mercato
ORDER_TYPE_BUY_LIMIT	Ordine pendente Buy Limit
ORDER_TYPE_SELL_LIMIT	Ordine pendente Sell Limit
ORDER_TYPE_BUY_STOP	Ordine pendente Buy Stop
ORDER_TYPE_SELL_STOP	Ordine pendente Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Al raggiungimento del prezzo dell'ordine, l'ordine pendente Buy Limit viene posto al prezzo StopLimit

ID	Descrizione
ORDER_TYPE_SELL_STOP_LIMIT	Al raggiungimento del prezzo dell'ordine, l'ordine pendente Sell Limit viene posto al prezzo StopLimit
ORDER_TYPE_CLOSE_BY	Ordine per la chiusura di una posizione con una posizione opposta

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
Stampa("Autore del pacchetto MetaTrader5:",MT5 .__ author__)
Stampa("Versione del pacchetto MetaTrader5:",MT5 .__ version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene valuta del conto
account_currency=mt5.account_info().currency
print("Valuta del conto:",account_currency)

# dispone la lista dei simboli
symbols=("EURUSD","GBPUSD","USDJPY", "USDCHF","EURJPY","GBPJPY")
print("Simboli col margine da controllare:", symbols)
action=mt5.ORDER_TYPE_BUY
lot=0.1
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"non trovato, saltato")
        continue
    if not symbol_info.visible:
        print(symbol, "se non visibile, provare ad attivarlo")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) fallito, saltato",symbol)
            continue
    ask=mt5.symbol_info_tick(symbol).ask
    margin=mt5.order_calc_margin(action,symbol,lot,ask)
    if margin != None:
        print("   {} buy {} margine lotto: {} {}".format(symbol,lot,margin,account_cu
    else:
        print("order_calc_margin fallito: , error code =", mt5.last_error())

# interrompe la connessione al terminale MetaTrader 5
```

```
mt5.shutdown()


Result:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD

Symbols to check margin: ('EURUSD', 'GBPUSD', 'USDJPY', 'USDCHF', 'EURJPY', 'GBPJPY')
   EURUSD buy 0.1 lot margin: 109.91 USD
   GBPUSD buy 0.1 lot margin: 122.73 USD
   USDJPY buy 0.1 lot margin: 100.0 USD
   USDCHF buy 0.1 lot margin: 100.0 USD
   EURJPY buy 0.1 lot margin: 109.91 USD
   GBPJPY buy 0.1 lot margin: 122.73 USD
```

See also

order_calc_profit, order_check

order_calc_profit

Restituisce il profitto nella valuta del conto per un'operazione di trading specificata.

```
order_calc_profit(
   action,          // tipo di ordine (ORDER_TYPE_BUY o ORDER_TYPE_SELL)
   symbol,          // nome del simbolo
   volume,          // volume
   price_open,      // prezzo di apertura
   price_close      // prezzo di chiusura
   );
```

Parametri

action

[in] Il tipo di ordine può richiedere uno dei due valori ORDER_TYPE di enumerazione: ORDER_TYPE_BUY o ORDER_TYPE_SELL. Parametro unnamed richiesto.

symbol

[in] Nome dello strumento finanziario. Parametro richiesto senza nome.

volume

[in] Volume delle operazioni di trading. Parametro richiesto senza nome.

price_open

[in] Prezzo di apertura. Parametro richiesto senza nome.

price_close

[in] Prezzo di chiusura. Parametro richiesto senza nome.

Valore di Ritorno

Valore Real in caso di successo, altrimenti Nessuno. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione consente di stimare un risultato dell'operazione di trading sull'attuale account e nell'ambiente di trading corrente. La funzione è simile a OrderCalcProfit.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
Stampa("Autore del pacchetto MetaTrader5:",MT5 .__ author__)
Stampa("Versione del pacchetto MetaTrader5:",MT5 .__ version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
```

```
    quit()

# ottiene la valuta dell'account
account_currency=mt5.account_info().currency
print("Valuta dell'account:",account_currency)

# dispone la lista dei simboli
symbols = ("EURUSD","GBPUSD","USDJPY")
print("Simboli da controllarne il margine:", symbols)
# profitto stimato per acquisti e vendite
lot=1.0
distance=300
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"non trovato, saltato")
        continue
    if not symbol_info.visible:
        print(symbol, "se non visibile, provare ad attivarlo")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) fallito, saltato",symbol)
            continue
    point=mt5.symbol_info(symbol).point
    symbol_tick=mt5.symbol_info_tick(symbol)
    ask=symbol_tick.ask
    bid=symbol_tick.bid
    buy_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_BUY,symbol,lot,ask,ask+distance*po
    if buy_profit!=None:
        print("   buy {} {} lot: profitto su {} punti => {} {}".format(symbol,lot,dist
    else:
        print("order_calc_profit(ORDER_TYPE_BUY) fallito, error code =",mt5.last_erro
    sell_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_SELL,symbol,lot,bid,bid-distance*
    if sell_profit!=None:
        print("   sell {} {} lots: profitto su {}punti => {} {}".format(symbol,lot,dis
    else:
        print("order_calc_profit(ORDER_TYPE_SELL) fallito, error code =",mt5.last_erro
    print()

# interrompe la connessione al terminale MetaTrader
mt5.shutdown()


 Risultato:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
Simboli per cui controllare il margine: ('EURUSD', 'GBPUSD', 'USDJPY')
   buy EURUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell EURUSD 1.0 lot: profit on 300 points => 300.0 USD

   buy GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
```

```
buy USDJPY 1.0 lot: profit on 300 points => 276.54 USD
sell USDJPY 1.0 lot: profit on 300 points => 278.09 USD
```

See also

order_calc_margin, order_check

order_check

Verificare la sufficienza dei fondi per eseguire un' operazione di trading richiesta. Il risultato del controllo viene restituito come struttura MqlTradeCheckResult.

```
order_check(
   request      // richiedi struttura
   );
```

Parametri

request

[in] MqlTradeRequest tipo struttura che descrive un'azione di trading richiesta. Parametro richiesto senza nome. Di seguito sono descritti esempi di compilazione di una richiesta ed il contenuto dell'enumerazione.

Valore di Ritorno

Controlla il risultato come struttura MqlTradeCheckResult. Il campo *request* nella risposta contiene la struttura di una richiesta di trading passata ad order_check().

Nota

L' invio riuscito di una richiesta **non implica che l'operazione di trading richiesta verrà eseguita con successo**. La funzione order_check è simile a OrderCheck.

TRADE_REQUEST_ACTIONS

ID	Descrizione
TRADE_ACTION_DEAL	Effettua un ordine per un affare(deal) istantaneo con i parametri specificati (imposta un market order)
TRADE_ACTION_PENDING	Piazza un ordine per eseguire un affare(deal) alle condizioni specificate (ordine pendente)
TRADE_ACTION_SLTP	Cambia lo Stop Loss ed il Take Profit della posizione aperta
TRADE_ACTION_MODIFY	Modifica i parametri dell'ordine di trading precedentemente piazzato
TRADE_ACTION_REMOVE	Rimuove l'ordine pendente precedentemente piazzato
TRADE_ACTION_CLOSE_BY	Chiude una posizione con una opposta

ORDER_TYPE_FILLING

ID	Descrizione
ORDER_FILLING_FOK	Questa politica di esecuzione indica che un ordine può essere eseguito solo nel volume specificato. Se la quantità necessaria di uno strumento finanziario non è attualmente disponibile nel mercato, l'ordine non verrà eseguito. Il volume desiderato può essere composto da diverse offerte disponibili.

ID	Descrizione
ORDER_FILLING_IOC	Un accordo per eseguire un affare(deal) al volume massimo disponibile sul mercato entro il volume specificato nell'ordine. Se la richiesta non può essere riempita completamente, verrà eseguito un ordine con il volume disponibile ed il volume rimanente verrà annullato.
ORDER_FILLING_RETURN	Questa politica è utilizzata solo per il mercato (ORDER_TYPE_BUY e ORDER_TYPE_SELL), ordini limit e stop limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT) e solo per i simboli con le modalità di esecuzione Market o Exchange. Se riempito parzialmente, un ordine di mercato o ordine limit, con il volume rimanente non viene annullato e viene ulteriormente elaborato. Durante l'attivazione degli ordini ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT, viene creato un ordine limit appropriato ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT con il tipo ORDER_FILLING_RETURN.

ORDER_TYPE_TIME

ID	Descrizione
ORDER_TIME_GTC	L'ordine rimane in coda fino a quando non viene annullato manualmente
ORDER_TIME_DAY	L'ordine è attivo solo durante il giorno di trading corrente
ORDER_TIME_SPECIFIED	L'ordine è attivo fino alla data specificata
ORDER_TIME_SPECIFIED_DAY	L'ordine è attivo fino alle 23:59:59 del giorno specificato. Se questo orario sembra essere fuori da una sessione di trading, la scadenza viene elaborata nell'orario di trading più vicino.

Esempio:

```
import MetaTrader5 as mt5
# mostra i dati sul package MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce una connessione al terminale di trading MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene la valuta dell'account
account_currency=mt5.account_info().currency
print("Valuta dell'account:",account_currency)
```

```
# prepara la struttura richiesta
symbol="USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "non trovato, non posso chiamare order_check()")
    mt5.shutdown()
    quit()

# se il simbolo non è disponibile in MarketWatch, aggiungerlo
if not symbol_info.visible:
    print(symbol, "non è visibile, prova ad attivarlo")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}) failed, exit",symbol)
        mt5.shutdown()
        quit()

# prepara la richiesta
point=mt5.symbol_info(symbol).point
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": 1.0,
    "type": mt5.ORDER_TYPE_BUY,
    "price": mt5.symbol_info_tick(symbol).ask,
    "sl": mt5.symbol_info_tick(symbol).ask-100*point,
    "tp": mt5.symbol_info_tick(symbol).ask+100*point,
    "deviation": 10,
    "magic": 234000,
    "comment": "python script",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# esegue il controllo e visualizza il risultato "così com'è"
result = mt5.order_check(request)
print(result);
# richiede il risultato come dizionario e lo visualizza elemento per elemento
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
    # se questa è una struttura di richiesta di trading, visualizzala anche elemento p
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}={}".format(tradereq_filed,traderequest_dict
```

```
# interrompe la connessione al terminale MetaTrader
mt5.shutdown()


 Risultato:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
   retcode=0
   balance=101300.53
   equity=68319.53
   profit=-32981.0
   margin=51193.67
   margin_free=17125.86
   margin_level=133.45308121101692
   comment=Done
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=1.0,
       traderequest: action=1
       traderequest: magic=234000
       traderequest: order=0
       traderequest: symbol=USDJPY
       traderequest: volume=1.0
       traderequest: price=108.081
       traderequest: stoplimit=0.0
       traderequest: sl=107.98100000000001
       traderequest: tp=108.181
       traderequest: deviation=10
       traderequest: type=0
       traderequest: type_filling=2
       traderequest: type_time=0
       traderequest: expiration=0
       traderequest: comment=python script
       traderequest: position=0
       traderequest: position_by=0
```

See also

order_send, OrderCheck, Tipi di operazioni di trading, Struttura richiesta di trading, Struttura dei risultati del controllo della richiesta di trading, Struttura dei risultati della richiesta di trading

order_send

Invia un richiesta per eseguire unoperazione di trading dal terminale al trade server. La funzione è simile ad OrderSend.

```
order_send(
   request      // richiedi struttura
   );
```

Parametri

request

[in] MqlTradeRequest tipo struttura che descrive un'azione di trading richiesta. Parametro richiesto senza nome. Di seguito sono descritti esempi di compilazione di una richiesta ed il contenuto dell'enumerazione.

Valore di Ritorno

Risultato dell'esecuzione come struttura MqlTradeResult. Il campo *request* nella risposta contiene la struttura di una richiesta di trading passata ad order_send().

La struttura della richiesta di trading MqlTradeRequest

Campo	Descrizione
action	Tipo di operazione di trading. Il valore può essere uno dei valori dell'enumerazione TRADE_REQUEST_ACTIONS
magic	EA ID. Consente di organizzare la gestione analitica degli ordini di trading. Ogni EA può impostare un ID univoco quando invia una richiesta di trading
order	Ticket dell'ordine. Richiesto per modificare gli ordini pendenti
symbol	Il nome dello strumento di trading, per il quale viene piazzato l'ordine. Non richiesto per modificare ordini e chiudere posizioni
volume	Volume richiesto di un affare(deal) in lotti. Un volume reale quando si fa un affare(deal) dipende dal tipo di esecuzione dell'ordine.
price	Prezzo al quale eseguire un ordine. Il prezzo non è fissato nel caso di ordini di mercato per strumenti di tipo "Esecuzione a Mercato" (SYMBOL_TRADE_EXECUTION_MARKET) avente il tipo TRADE_ACTION_DEAL
stoplimit	Un prezzo a cui viene impostato un ordine limit pendente quando il prezzo raggiunge il valore 'price' (questa condizione è obbligatoria). L'ordine pendente non viene passato al sistema di trading fino a quel momento
sl	Un prezzo per cui viene attivato un ordine Stop Loss quando il prezzo si sposta in una direzione sfavorevole
tp	Un prezzo per cui viene attivato un ordine Take Profit quando il prezzo si muove in una direzione favorevole
deviation	Deviazione massima accettabile dal prezzo richiesto, specificata in punti

Campo	Descrizione
type	Tipo di ordine. Il valore può essere uno dei valori dell'enumerazione ORDER_TYPE
type_filling	Tipo di riempimento dell'ordine. Il valore può essere uno dei valori ORDER_TYPE_FILLING
type_time	Tipo di ordine per scadenza. Il valore può essere uno dei valori ORDER_TYPE_TIME
expiration	Tempo di espirazione(scadenza9 dell'ordine pendente (per tipo di ordini TIME_SPECIFIED)
comment	Commento ad un ordine
position	Ticket della posizione. Lo riempie quando si cambia e chiude una posizione per la sua chiara identificazione. Di solito, è lo stesso del ticket dell'ordine che ha aperto la posizione.
position_by	Ticket della posizione opposta. Viene utilizzato quando si chiude una posizione da una posizione opposta (aperta con lo stesso simbolo ma nella direzione opposta).

Nota

Una richiesta di trading supera diverse fasi di verifica sul trade server. Innanzitutto, viene controllata la validità di tutti i campi della *richiesta* necessari. Se non ci sono errori, il server accetta l'ordine per l'ulteriore gestione. Vedere la descrizione della funzione OrderSend per i dettagli sull'esecuzione delle operazioni di trading.

Esempio:

```
import time
import MetaTrader5 as mt5

# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ", mt5.__author__)
print("MetaTrader5 package version: ", mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# prepara la struttura della richiesta di buy
symbol = "USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "non trovato, non posso chiamare order_check()")
    mt5.shutdown()
    quit()

# se il simbolo non è disponibile in MarketWatch, aggiungerlo
if not symbol_info.visible:
    print(symbol, "non è visibile, prova ad attivarlo")
```

```
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()

lot = 0.1
point = mt5.symbol_info(symbol).point
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_BUY,
    "price": price,
    "sl": price - 100 * point,
    "tp": price + 100 * point,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# invia una richiesta di trading
result = mt5.order_send(request)
# controlla il risultato dell'esecuzione
print("1. order_send(): by {} {} lots at {} with deviation={} points".format(symbol,lo
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("2. order_send fallito, retcode={}".format(result.retcode))
   # richiede il risultato come dizionario e lo visualizza elemento per elemento
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # se questa è una struttura di richiesta di trading, visualizzala anche elemen
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_
    print("shutdown() and quit")
    mt5.shutdown()
    quit()

print("2. order_send fatto, ", result)
print("   aperta posizione con POSITION_TICKET={}".format(result.order))
print("   sleep per 2 secondi prima di chiudere la posizione #{}".format(result.order)
time.sleep(2)
# crea una richiesta di chiusura
position_id=result.order
```

```
price=mt5.symbol_info_tick(symbol).bid
deviation=20
request={
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_SELL,
    "position": position_id,
    "price": price,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script close",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}
# invia una richiesta di trading
result=mt5.order_send(request)
# controlla il risultato dell'esecuzione
print("3. chiudi pozione #{}: sell {} {} lotti a {} con deviazione={} punti".format(p
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("4. order_send fallito, retcode={}".format(result.retcode))
    print("   risultato",result)
else:
    print("4. posizione #{} chiusa, {}".format(position_id,result))
   # richiede il risultato come dizionario e lo visualizza elemento per elemento
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # se questa è una struttura di richiesta di trading, visualizzala anche elemen
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

```
Risultato
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29
1. order_send(): di USDJPY 0.1 lotti a 108.023 con deviazione=20 punti
2. order_send fatto,  OrderSendResult(retcode=10009, deal=535084512, order=557416535,
  aperta posizione con POSITION_TICKET=557416535
  sleep per 2 secondi prima di chiudere la pozione #557416535
3. chiudi posizione #557416535: sell USDJPY 0.1 lotti a 108.018 con deviazione=20 punt
4. posizione #557416535 chiusa, OrderSendResult(retcode=10009, deal=535084631, order=5
   retcode=10009
   deal=535084631
   order=557416654
```

```
    volume=0.1
    price=108.015
    bid=108.015
    ask=108.02
    comment=Request executed
    request_id=55
    retcode_external=0
    request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=0.1,
        traderequest: action=1
        traderequest: magic=234000
        traderequest: order=0
        traderequest: symbol=USDJPY
        traderequest: volume=0.1
        traderequest: price=108.018
        traderequest: stoplimit=0.0
        traderequest: sl=0.0
        traderequest: tp=0.0
        traderequest: deviation=20
        traderequest: type=1
        traderequest: type_filling=2
        traderequest: type_time=0
        traderequest: expiration=0
        traderequest: comment=python script close
        traderequest: position=557416535
        traderequest: position_by=0
```

See also

order_check, OrderSend, Tipi di operazioni di trading, Struttura richiesta di trading, Struttura dei risultati del controllo della richiesta di trading, Struttura dei risultati della richiesta di trading

positions_total

Ottiene il numero di posizioni aperte.

```
positions_total()
```

Valore di Ritorno

Valore Integer.

Nota

La funzione è simile a PositionsTotal.

Esempio:

```
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# verifica la presenza di posizioni aperte
positions_total=mt5.positions_total()
if positions_total>0:
    print("Totale posizioni=",positions_total)
else:
    print("Posizioni non trovate")

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

positions_get, orders_total

positions_get

Ottiene posizioni aperte con la possibilità di filtrare per simbolo o ticket. Esistono tre opzioni di chiamata.

Chiamata senza parametri. Restituisce posizioni open per tutti i simboli.

```
positions_get()
```

Chiama specificando un simbolo per cui devono essere ricevute posizioni aperte.

```
positions_get(
   symbol="SYMBOL"      // nome del simbolo
)
```

Chiama specificando un gruppo di simboli per cui devono essere ricevute le posizioni aperte.

```
positions_get(
   group="GROUP"        // filtra per la selezione delle posizioni tramite simboli
)
```

Chiama specificando un ticket di posizione.

```
positions_get(
   ticket=TICKET        // ticket
)
```

Parametri

symbol="SYMBOL"

[in] Nome simbolo. Parametro named opzionale. Se viene specificato un simbolo, il parametro *ticket* viene ignorato.

group="GROUP"

[in] Il filtro per disporre un gruppo di simboli necessari. Parametro named opzionale. Se viene specificato il gruppo, la funzione restituisce solo le posizioni che soddisfano un criterio specificato per un nome simbolo.

ticket=TICKET

[in] Ticket posizione (POSITION_TICKET). Parametro named facoltativo.

Valore di Ritorno

Restituisce informazioni sotto forma di una struttura denominata tupla (namedtuple). Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione consente di ricevere tutte le posizioni aperte all'interno di una chiamata similmente al tandem PositionsTotal e PositionSelect.

Il parametro *group* può contenere diverse condizioni separate da virgola. Una condizione può essere impostata come maschera usando '*'. Il simbolo di negazione logica "!" può essere utilizzato per un'esclusione. Tutte le condizioni vengono applicate in sequenza, il che significa che le condizioni di inclusione in un gruppo devono essere specificate prima, seguite da una condizione di esclusione.

Per esempio, *group*= "*, !EUR" significa che le posizioni per tutti i simboli devono essere selezionate per prime e quelle che contengono "EUR" nei nomi dei simboli devono essere escluse in seguito.

Esempio:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # larghezza massima della tabella da visuali
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene posizioni aperte su USDCHF
positions=mt5.positions_get(symbol="USDCHF")
if positions==None:
    print("Niente posizioni su USDCHF, error code={}".format(mt5.last_error()))
elif len(positions)>0:
    print("Totale posizioni su USDCHF =",len(positions))
    # mostra tutte le posizioni aperte
    for position in positions:
        print(position)

# ottiene l'elenco delle posizioni sui simboli i cui nomi contengono "*USD*"
usd_positions=mt5.positions_get(group="*USD*")
if usd_positions==None:
    print("Niente posizioni col gruppo=\"*USD*\", error code={}".format(mt5.last_erro
elif len(usd_positions)>0:
    print("positions_get(group=\"*USD*\")={}".format(len(usd_positions)))
    # visualizza queste posizioni come una tabella usando pandas.DataFrame
    df=pd.DataFrame(list(usd_positions),columns=usd_positions[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    df.drop(['time_update', 'time_msc', 'time_update_msc', 'external_id'], axis=1, in
    print(df)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()


Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

positions_get(group="*USD*")=5
```

	ticket	time	type	magic	identifier	reason	volume	price_open
0	548297723	2020-03-18 15:00:55	1	0	548297723	3	0.01	1.09301
1	548655158	2020-03-18 20:31:26	0	0	548655158	3	0.01	1.08676
2	548663803	2020-03-18 20:40:04	0	0	548663803	3	0.01	1.08640
3	548847168	2020-03-19 01:10:05	0	0	548847168	3	0.01	1.09545
4	548847194	2020-03-19 01:10:07	0	0	548847194	3	0.02	1.09536

See also

positions_total, orders_get

history_orders_total

Ottiene il numero di ordini nella cronistoria di trading entro l'intervallo specificato.

```
history_orders_total(
   date_from,     // data da cui vengono richiesti gli ordini
   date_to        // data, fino alla quale sono richiesti gli ordini
   )
```

Parametri

date_from

[in] Data di richiesta degli ordini. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

date_to

[in] Data, fino alla quale sono richiesti gli ordini. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

Valore di Ritorno

Valore Integer.

Nota

La funzione è simile a HistoryOrdersTotal.

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene il numero di ordini nella cronistoria
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_total(from_date, datetime.now())
if history_orders>0:
    print("Totale ordini cronistorici =",history_orders)
else:
    print("Ordini non trovati nella cronistoria")

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

history_orders_get, history_deals_total

history_orders_get

Ottiene ordini dalla cronistoria di trading con la possibilità di filtrare per ticket o posizione. Esistono tre opzioni di chiamata.

Chiamata specificando un intervallo di tempo. Restituisce tutti gli ordini che rientrano nell'intervallo specificato.

```
history_orders_get(
   date_from,                // data da cui vengono richiesti gli ordini
   date_to,                  // data, fino alla quale sono richiesti gli ordini
   group="GROUP"         // filtro per la selezione degli ordini tramite simboli
   )
```

Chiama specificando il ticket dell'ordine. Restituisce un ordine con il ticket specificato.

```
history_orders_get(
   ticket=TICKET         // ticket dell'ordine
)
```

Chiama specificando il ticket della posizione. Restituisce tutti gli ordini con il ticket della posizione specificato nella proprietà ORDER_POSITION_ID.

```
history_orders_get(
   position=POSITION     // ticket della positione
)
```

Parametri

date_from

[in] Data di richiesta degli ordini. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Il parametro richiesto unnamed viene specificato per primo.

date_to

[in] Data, fino alla quale sono richiesti gli ordini. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Il parametro richiesto senza nome viene specificato per secondo.

group="GROUP"

[in] Il filtro per disporre un gruppo di simboli necessari. Parametro named opzionale. Se viene specificato il gruppo, la funzione restituisce solo gli ordini che soddisfano un criterio specificato per un nome simbolo.

ticket=TICKET

[in] Ticket dell'ordine che dovrebbe essere ricevuto. Parametro opzionale. Se non specificato, il filtro non viene applicato.

position=POSITION

[in] Ticket di una posizione (memorizzato in ORDER_POSITION_ID) per cui devono essere ricevuti tutti gli ordini. Parametro opzionale. Se non specificato, il filtro non viene applicato.

Valore di Ritorno

Restituisce informazioni sotto forma di una struttura denominata tupla (namedtuple). Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione consente di ricevere tutti gli ordini della cronistoria entro un periodo specificato in una singola chiamata simile al tandem HistoryOrdersTotal e HistoryOrderSelect.

Il parametro *group* può contenere diverse condizioni separate da virgola. Una condizione può essere impostata come maschera usando '*'. Il simbolo di negazione logica "!" può essere utilizzato per un'esclusione. Tutte le condizioni vengono applicate in sequenza, il che significa che le condizioni di inclusione in un gruppo devono essere specificate prima, seguite da una condizione di esclusione. Ad esempio, *group*= "*, !EUR" significa che i deals per tutti i simboli dovrebbero essere selezionati per primi e quelli che contengono "EUR" nei nomi dei simboli dovrebbero essere esclusi in seguito.

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # larghezza massima della tabella da visuali
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene il numero di ordini nella cronistoria
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_get(from_date, to_date, group="*GBP*")
if history_orders==None:
    print("Nessun ordine della cronistoria con gruppo=\"*GBP*\", error code={}".format
elif len(history_orders)>0:
    print("history_orders_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,le
print()

# mostra tutti gli ordini storici per un ticket di posizione
position_id=530218319
position_history_orders=mt5.history_orders_get(position=position_id)
if position_history_orders==None:
    print("Nessun ordine con posizione #{}".format(position_id))
    print("error code =",mt5.last_error())
elif len(position_history_orders)>0:
    print("Totale ordini della cronistoria sulla posizione #{}: {}".format(position_id
   # visualizza tutti gli ordini storici con il ticket della posizione specificata
```

```
    for position_order in position_history_orders:
        print(position_order)
    print()
    # visualizza questi ordini come una tabella usando pandas.DataFrame
    df=pd.DataFrame(list(position_history_orders),columns=position_history_orders[0]._
    df.drop(['time_expiration','type_time','state','position_by_id','reason','volume_
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    df['time_done'] = pd.to_datetime(df['time_done'], unit='s')
    print(df)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()

Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

history_orders_get(2020-01-01 00:00:00, 2020-03-25 17:17:32.058795, group="*GBP*")=14

Ordini storici totali nella posizione #530218319: 2
TradeOrder(ticket=530218319, time_setup=1582282114, time_setup_msc=1582282114681, time
TradeOrder(ticket=535548147, time_setup=1583176242, time_setup_msc=1583176242265, time

      ticket          time_setup  time_setup_msc           time_done  time_done_msc  t
0  530218319 2020-02-21 10:48:34   1582282114681 2020-02-21 16:49:37  1582303777582
1  535548147 2020-03-02 19:10:42   1583176242265 2020-03-02 19:10:42  1583176242265
```

See also

history_deals_total, history_deals_get

history_deals_total

Ottiene il numero di deals nella cronistoria di trading entro l'intervallo specificato.

```
history_deals_total(
   date_from,     // data in cui sono richiesti i deals
   date_to        // data, fino alla quale sono richiesti i deals
   )
```

Parametri

date_from

[in] Data di richiesta dei deals. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

date_to

[in] Data, fino alla quale sono richiesti i deals. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Parametro richiesto senza nome.

Valore di Ritorno

Valore Integer.

Nota

La funzione è simile a HistoryDealsTotal.

Esempio:

```
from datetime import datetime
import MetaTrader5 as mt5
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene il numero di deals nella cronistoria
from_date=datetime(2020,1,1)
to_date=datetime.now()
deals=mt5.history_deals_total(from_date, to_date)
if deals>0:
    print("Deals totali=",deals)
else:
    print("Deals non trovati nella cronistoria")

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()
```

See also

history_deals_get, history_orders_total

history_deals_get

Ottiene affari(deals) dalla cronistoria di trading entro l'intervallo specificato con la possibilità di filtrare per ticket o posizione.

Chiamata specificando un intervallo di tempo. Restituisce tutti i deals che rientrano nell'intervallo specificato.

```
history_deals_get(
   date_from,                // data in cui sono richiesti i deals
   date_to,                  // data, fino alla quale sono richiesti i deals
   group="GROUP"          // filtro per selezionare deals per simboli
   )
```

Chiama specificando il ticket dell'ordine. Restituisce tutti i deals con il ticket ordine specificato nelle proprietà DEAL_ORDER.

```
history_deals_get(
   ticket=TICKET          // ticket dell'ordine
)
```

Chiama specificando il ticket della posizione. Restituisce tutti i deals con il ticket della posizione specificato nella proprietà DEAL_POSITION_ID.

```
history_deals_get(
   position=POSITION     // ticket della positione
)
```

Parametri

date_from

[in] Data di richiesta degli ordini. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Il parametro richiesto unnamed viene specificato per primo.

date_to

[in] Data, fino alla quale sono richiesti gli ordini. Impostata dall'oggetto 'datetime' o come numero di secondi trascorsi dal 1970.01.01. Il parametro richiesto senza nome viene specificato per secondo.

group="GROUP"

[in] Il filtro per disporre un gruppo di simboli necessari. Parametro named opzionale. Se viene specificato il gruppo, la funzione restituisce solo gli affati che soddisfano un criterio specificato per un nome simbolo.

ticket=TICKET

[in] Ticket di un ordine (archiviato inDEAL_ORDER) per cui devono essere ricevuti tutti i deals. Parametro opzionale. Se non specificato, il filtro non viene applicato.

position=POSITION

[in] Ticket di una posizione (memorizzato in DEAL_POSITION_ID) per cui devono essere ricevuti tutti i deals. Parametro opzionale. Se non specificato, il filtro non viene applicato.

Valore di Ritorno

Restituisce informazioni sotto forma di una struttura denominata tupla (namedtuple). Restituisce Nessuna in caso di errore. Le informazioni sull'errore possono essere ottenute utilizzando last_error().

Nota

La funzione consente di ricevere tutti i deals della cronistoria entro un periodo specificato in una singola chiamata simile al tandem HistoryDealsTotal e HistoryDealSelect.

Il parametro *group* consente di ordinare i deals in base ai simboli. '*' può essere usato all'inizio e alla fine di una stringa.

Il parametro *group* può contenere diverse condizioni separate da virgola. Una condizione può essere impostata come maschera usando '*'. Il simbolo di negazione logica "!" può essere utilizzato per un'esclusione. Tutte le condizioni vengono applicate in sequenza, il che significa che le condizioni di inclusione in un gruppo devono essere specificate prima, seguite da una condizione di esclusione. Ad esempio, *group*= "*, !EUR" significa che i deals per tutti i simboli dovrebbero essere selezionati per primi e quelli che contengono "EUR" nei nomi dei simboli dovrebbero essere esclusi in seguito.

Esempio:

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # numero di colonne da visualizzare
pd.set_option('display.width', 1500)      # larghezza massima della tabella da visuali
# visualizza i dati sul pacchetto MetaTrader 5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# stabilisce la connessione al terminale MetaTrader 5
if not mt5.initialize():
    print("initialize() fallito, error code =",mt5.last_error())
    quit()

# ottiene il numero di deals nella cronistoria
from_date=datetime(2020,1,1)
to_date=datetime.now()
# ottiene deals per simboli i cui nomi contengono "GBP" entro un intervallo specificat
deals=mt5.history_deals_get(from_date, to_date, group="*GBP*")
if deals==None:
    print("Nessun deals per il gruppo=\"*USD*\", error code={}".format(mt5.last_error
elif len(deals)> 0:
    print("history_deals_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,len

# ottiene deals per simboli i cui nomi non contengono né "EUR" né "GBP"
deals = mt5.history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*")
if deals == None:
    print("Nessun deal, error code={}".format(mt5.last_error()))
elif len(deals) > 0:
    print("history_deals_get(from_date, to_date, group=\"*,!*EUR*,!*GBP*\") =", len(de
```

```
    # mostra tutti i deals ottenuti 'così come sono'<
    for deal in deals:
        print("  ",deal)
    print()
    # mostra questi deals come una tabella usando pandas.DataFrame
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# ottiene tutti i deals relativi alla posizione #530218319
position_id=530218319
position_deals = mt5.history_deals_get(position=position_id)
if position_deals == None:
    print("Nessun deal con la posizione #{}".format(position_id))
    print("error code =", mt5.last_error())
elif len(position_deals) > 0:
    print("Deals con id posizione #{}: {}".format(position_id, len(position_deals)))
    # mostra questi deals come una tabella usando pandas.DataFrame
    df=pd.DataFrame(list(position_deals),columns=position_deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)

# interrompe la connessione al terminale MetaTrader 5
mt5.shutdown()

Risultato:
Autore del pacchetto MetaTrader5: MetaQuotes Software Corp.
Versione del pacchetto MetaTrader5: 5.0.29

history_deals_get(from_date, to_date, group="*GBP*") = 14

history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*") = 7
   TradeDeal(ticket=506966741, order=0, time=1582202125, time_msc=1582202125419, type=
   TradeDeal(ticket=507962919, order=530218319, time=1582303777, time_msc=158230377758
   TradeDeal(ticket=513149059, order=535548147, time=1583176242, time_msc=158317624226
   TradeDeal(ticket=516943494, order=539349382, time=1583510003, time_msc=158351000389
   TradeDeal(ticket=516943915, order=539349802, time=1583510025, time_msc=158351002505
   TradeDeal(ticket=517139682, order=539557870, time=1583520201, time_msc=158352020122
   TradeDeal(ticket=517139716, order=539557909, time=1583520202, time_msc=158352020297

      ticket      order                time       time_msc  type  entry  magic  positi
0  506966741          0 2020-02-20 12:35:25  1582202125419     2      0      0
1  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
2  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
3  516943494  539349382 2020-03-06 15:53:23  1583510003895     1      0      0    5393
4  516943915  539349802 2020-03-06 15:53:45  1583510025054     0      0      0    5393
5  517139682  539557870 2020-03-06 18:43:21  1583520201227     0      1      0    5393
6  517139716  539557909 2020-03-06 18:43:22  1583520202971     1      1      0    5393
```

```
Deals con ID posizione #530218319: 2
      ticket      order                time       time_msc  type  entry  magic  positi
0  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
1  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
```

See also

history_deals_total, history_orders_get

I Modelli ONNX nell'Apprendimento Automatico

ONNX (Open Neural Network Exchange) è un formato open source per i modelli di apprendimento automatico. Questo progetto presenta diversi vantaggi:

- ONNX è supportato da grandi aziende come Microsoft, Facebook, Amazon e altri partner.
- Il suo formato aperto consente le conversioni di formato tra diversi kit di strumenti di apprendimento automatico, mentre Microsoft ONNXMLTools permette di convertire i modelli nel formato ONNX.
- MQL5 fornisce la conversione automatica del tipo di dati per gli ingressi e le uscite del modello se il tipo di parametro passato non corrisponde al modello.
- I modelli ONNX possono essere creati utilizzando vari strumenti di apprendimento automatico. Attualmente sono supportati in Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch e OpenCV. Sono disponibili anche interfacce per altri framework e librerie popolari.
- Con il linguaggio MQL5, è possibile implementare un modello ONNX in una strategia di trading e utilizzarlo insieme a tutti i vantaggi della piattaforma MetaTrader 5 per operazioni efficienti nei mercati finanziari.
- Prima di mettere a punto un modello per il trading dal vivo, puoi testare il comportamento del modello sui dati storici nello Strategy Tester, senza utilizzare strumenti di terze parti.

MQL5 fornisce le seguenti funzioni per lavorare con ONNX:

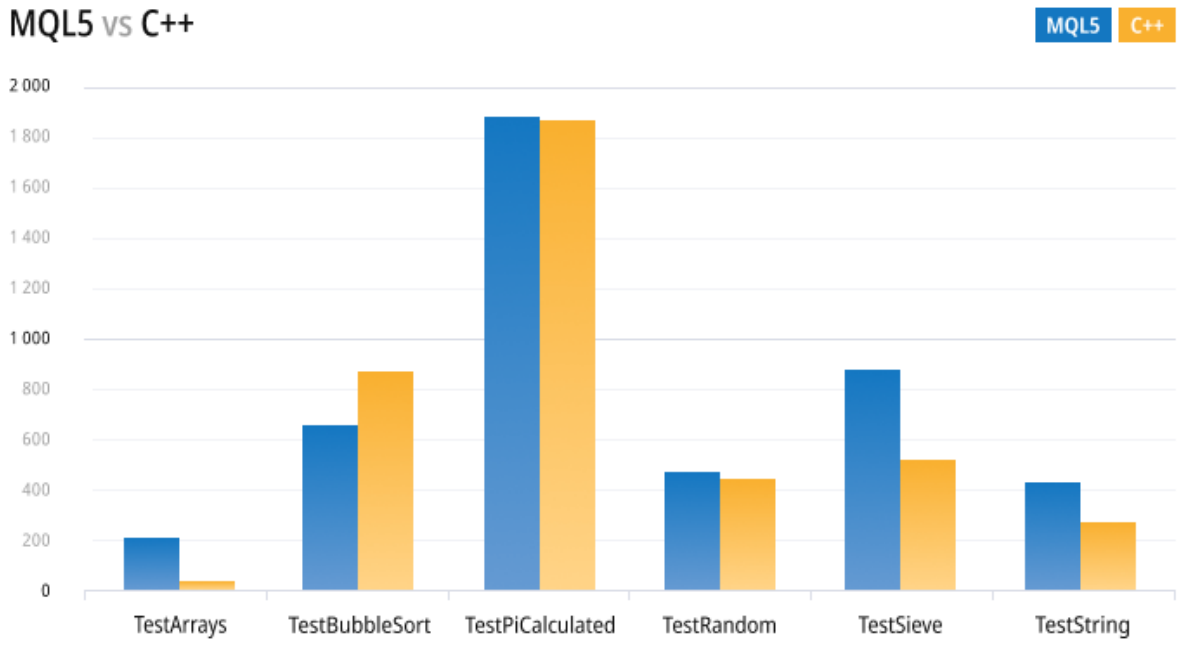
Funzione	Azione
OnnxCreate	Crea una sessione ONNX, caricando un modello da un file *.onnx
OnnxCreateFromBuffer	Crea una sessione ONNX, caricando un modello da un array di dati
OnnxRelease	Chiude una sessione ONNX
OnnxRun	Esegue un modello ONNX
OnnxGetInputCount	Ottiene il numero di ingressi in un modello ONNX
OnnxGetOutputCount	Ottiene il numero di uscite in un modello ONNX
OnnxGetInputName	Ottiene il nome dell'ingresso di un modello tramite indice
OnnxGetOutputName	Ottiene il nome dell'uscita di un modello tramite indice
OnnxGetInputTypeInfo	Ottiene la descrizione del tipo di ingresso dal modello
OnnxGetOutputTypeInfo	Ottiene la descrizione del tipo di uscita dal modello
OnnxSetInputShape	Imposta la dimensione dei dati d'ingresso di un modello tramite indice
OnnxSetOutputShape	Imposta la dimensione dei dati di uscita di un modello tramite indice

Supporto ONNX in MQL5

ONNX è un formato aperto creato per rappresentare modelli di apprendimento automatico. Questo standard definisce un insieme comune di operatori e un formato di file comune per consentire agli sviluppatori di utilizzare modelli con framework, strumenti, runtime e compilatori diversi.

Pertanto, il formato aperto ONNX consente di ricevere e trasferire modelli di apprendimento automatico tra diverse piattaforme e kit di strumenti per l'apprendimento automatico. ONNX è implementato nel linguaggio MQL5 per consentire agli sviluppatori di AI di eseguire modelli creati nell'ambiente di esecuzione ad alte prestazioni fornito dalla piattaforma MetaTrader 5.

La velocità di esecuzione MQL5 è paragonabile a quella delle applicazioni C++. Questo è provato dai risultati di esecuzione di test standard su MQL5 e C++. Più piccola è la barra, meno tempo (in millisecondi) viene speso per l'esecuzione e migliore è il risultato. I test sono stati condotti su Windows 10 (build 17763) x64, Xeon E5-2630 v4 @ 2.20GHz, Memory: 65457 Mb.



Le nuove operazioni di trading asincrono e il supporto nativo ONNX offrono nuove opportunità che in precedenza erano disponibili solo per una manciata di sviluppatori di IA professionali e trader istituzionali. Il supporto ONNX in MQL5 consente ai trader di addestrare modelli per il trading sui mercati finanziari nel loro ambiente di sviluppo preferito e quindi di operare con bassi costi di rete, elevata velocità di aggiornamento dell'order book e invio asincrono degli ordini.

Attualmente, ONNX è sviluppato e gestito da aziende partner come Microsoft, Facebook, Amazon e altri, che garantiscono l'ulteriore sviluppo di questo progetto aperto.

Conversione Di Formato

ONNX è un formato aperto, che consente l'utilizzo di modelli da diversi kit di apprendimento automatico. Questo formato è supportato da molti framework, tra cui Chainer, Caffee2 e PyTorch.

Uno degli strumenti più popolari per la conversione di modelli in formato ONNX è Microsoft ONNXMLTools.

Le istruzioni per l'installazione e l'uso di ONNXMLTools sono disponibili presso il repo GitHub. Sono attualmente supportati i seguenti kit di strumenti:

- Keras (un wrapper di keras2onnx converter)
- Tensorflow (un wrapper di tf2onnx converter)
- scikit-learn (un wrapper di skl2onnx converter)
- Apple Core ML
- Spark ML (sperimentale)
- LightGBM
- libscm;
- XGBoost;
- H2O
- CatBoost

ONNXMLTools può essere facilmente installato. Per i dettagli di installazione e gli esempi di conversione del modello, vedere la pagina del progetto su https://github.com/onnx/onnxmltools#install.

Conversione automatica dei valori d'ingresso e d'uscita durante l'esecuzione di modelli ONNX

L'attuale versione ONNX in MQL5 supporta solo tensori per valori ingresso/uscita. I tensori sono array di dati con gli elementi dei seguenti tipi di dati:

Tipo ONNX	Corrisponde al tipo MQL5
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_UINT8	uchar
ONNX_DATA_TYPE_INT8	char
ONNX_DATA_TYPE_UINT16	ushort
ONNX_DATA_TYPE_INT16	short
ONNX_DATA_TYPE_INT32	int
ONNX_DATA_TYPE_INT64	long
ONNX_DATA_TYPE_FLOAT16	—
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint
ONNX_DATA_TYPE_UINT64	ulong
ONNX_DATA_TYPE_COMPLEX64	—
ONNX_DATA_TYPE_COMPLEX128	complex
ONNX_DATA_TYPE_BFLOAT16	—
ONNX_DATA_TYPE_STRING	—

Solo gli array, vettori e matrici (ci riferiremo a loro come **Data**) possono essere inseriti nei modelli ONNX come valori di ingresso/uscita.

Se i tipi di parametro non corrispondono al tipo di parametro del modello ONNX e viene chiamata OnnxRun senza il flag ONNX_NO_CONVERSION specificato, verrà applicata la conversione automatica dei dati. L'autoconversione implica che prima di eseguire un modello ONNX, l'operatore **Data** verrà copiato nei tensori ONNX con la relativa conversione.

Quando un modello ONNX viene eseguito senza autoconversione, il modello verrà calcolato utilizzando **Data** senza alcuna copia aggiuntiva.

> **IMPORTANTE!** L'autoconversione non controlla l'overflow (troncare), pertanto è necessario monitorare attentamente i dati e i tipi di dati immessi nel modello ONNX.

L'autoconversione supporta i seguenti tipi ONNX:

- ONNX_DATA_TYPE_BOOL
- ONNX_DATA_TYPE_FLOAT
- ONNX_DATA_TYPE_UINT8
- ONNX_DATA_TYPE_INT8
- ONNX_DATA_TYPE_UINT16
- ONNX_DATA_TYPE_INT16
- ONNX_DATA_TYPE_INT32
- ONNX_DATA_TYPE_INT64
- ONNX_DATA_TYPE_FLOAT16
- ONNX_DATA_TYPE_DOUBLE
- ONNX_DATA_TYPE_UINT32
- ONNX_DATA_TYPE_UINT64
- ONNX_DATA_TYPE_COMPLEX64
- ONNX_DATA_TYPE_COMPLEX128

Tipi non supportati:

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

Regole di Autoconversione per Tipi di Tensori

Se il tipo MQL5 non è incluso nella lista dei tipi supportati dal modello, l'esecuzione del modello ONNX restituirà l'errore ERR_ONNX_NOT_SUPPORTED (codice errore 5802).

Nota: Durante l'autoconversione, il tipo color viene processato come uint, mentre datetime viene processato come long.

Autoconversione dei valori in ingresso

Tipo ONNX (tipo elemento tensore)	Tipo MQL5 supportato da autoconversione
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Durante la conversione, gli elementi **Data** sono controllati da un semplice confronto con 0
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	Vedere ONNX_DATA_TYPE_FLOAT

Tipo ONNX (tipo elemento tensore)	Tipo MQL5 supportato da autoconversione
ONNX_DATA_TYPE_UINT16	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Autoconversione dei valori in uscita

Tipo ONNX (tipo elemento tensore)	Tipo MQL5 supportato da autoconversione
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Se l'elemento tensore è zero, l'elemento Data è impostato su 0; altrimenti il valore è 1
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT16	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	Vedere ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Vedi anche

Type Casting

Creazione di un Modello

Sono disponibili diversi metodi per ottenere un modello pronto nel formato ONNX. La popolare libreria ONNX Model Zoo contiene diversi modelli ONNX pre-addestrati per diversi tipi di attività. Il vantaggio di questa collezione è che il notebook di ogni modello contiene collegamenti al set di dati di addestramento e riferimenti al documento originale che descrive l'architettura del modello.

La maggior parte dei framework di apprendimento automatico usa Python. Per installare il runtime ONNX per Python, usare uno dei seguenti comandi:

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

Per richiamare il runtime ONNX in Python, usare il seguente comando:

```
import onnxruntime
session = onnxruntime.InferenceSession("path to model")
```

Per il modello d'ingresso e d'uscita, consultare la documentazione del modello pertinente. È anche possibile utilizzare strumenti di visualizzazione per visualizzare il modello, come Netron o WinML Dashboard. Nel runtime ONNX, puoi anche interrogare i metadati di un modello e i suoi ingressi e uscite:

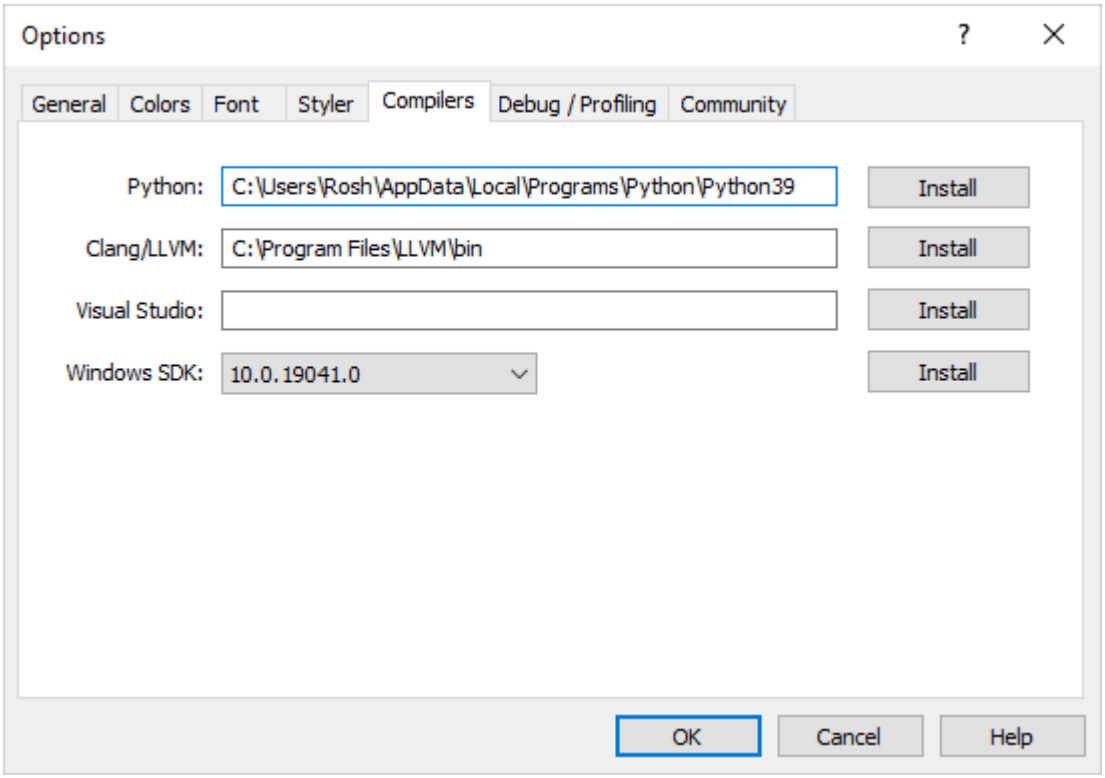
```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

È possibile creare modelli ONNX direttamente nel terminale MetaTrader 5 o in MetaEditor utilizzando Python.

Python in MetaTrader 5

MetaTrader 5 fornisce supporti preconfigurati per gli script Python. Per abilitare queste operazioni, gli sviluppatori del terminale forniscono il modulo MetaTrader5 per Python: https://pypi.org/project/MetaTrader5.

Nell'ambiente di sviluppo integrato Metaeditor, oltre a creare applicazioni in MQL5, è anche possibile eseguire script Python direttamente dall'editor. Per fare questo, specificare il percorso dell'eseguibile in Metaeditor settings:



Se Python non è installato sul computer, fare clic su Installa per scaricare il file di installazione.

È possibile creare uno script Python in MetaEdtior o caricarlo nella cartella dati del terminale ed eseguirlo immediatamente utilizzando la chiave F7 (Compila). Questo aprirà il terminale MetaTrader 5 con lo script in esecuzione sul grafico corrente. I messaggi dalla console Python (stdout, stderr) saranno visualizzati sotto la sezione Errori.

Operazioni con i modelli in MetaTrader 5

Il linguaggio MQL5 consente di eseguire i modelli ONNX direttamente nel terminale MetaTrader 5. Questo viene fatto in tre passaggi:

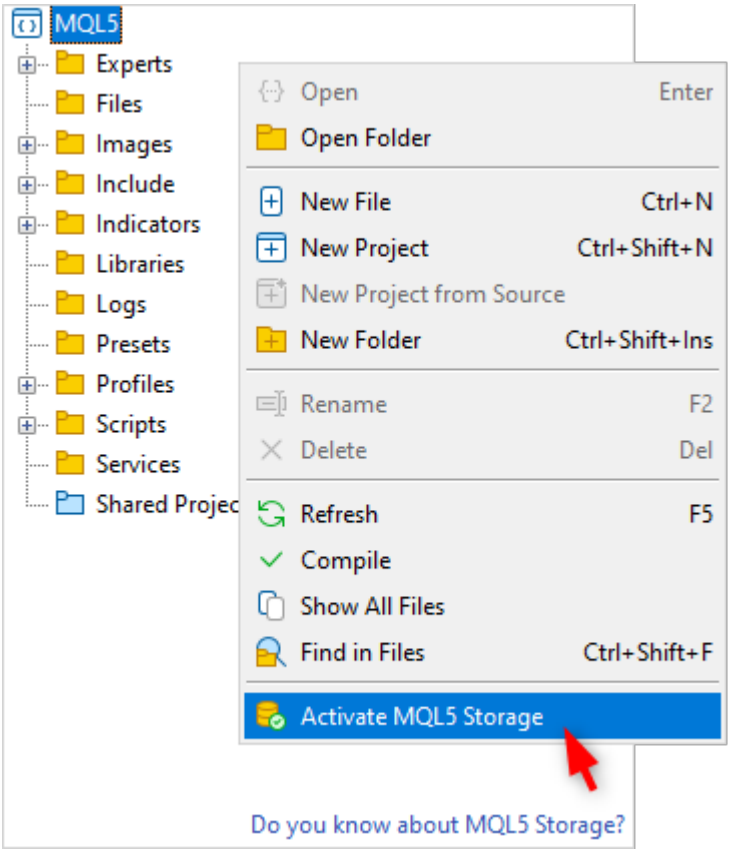
1. Addestra il modello in una piattaforma di terze parti, come Python.
2. Convertire il modello in ONNX.
3. Includere il modello ONNX in un Expert Advisor utilizzando le funzioni ONNX ed eseguire nel terminale MetaTrader 5.

L'ntegrazione di Python in MQL5 consente di eseguire uno script python e salvare un modello ONNX nel Metaeditor o eseguirlo direttamente su un grafico in MetaTrader 5. È possibile addestrare il modello utilizzando uno script Python pre-scritto tutte le volte che è necessario direttamente nel terminale. La libreria include funzioni pronte all'uso per ottenere i dati sui prezzi, che possono essere inseriti in un modello ONNX:

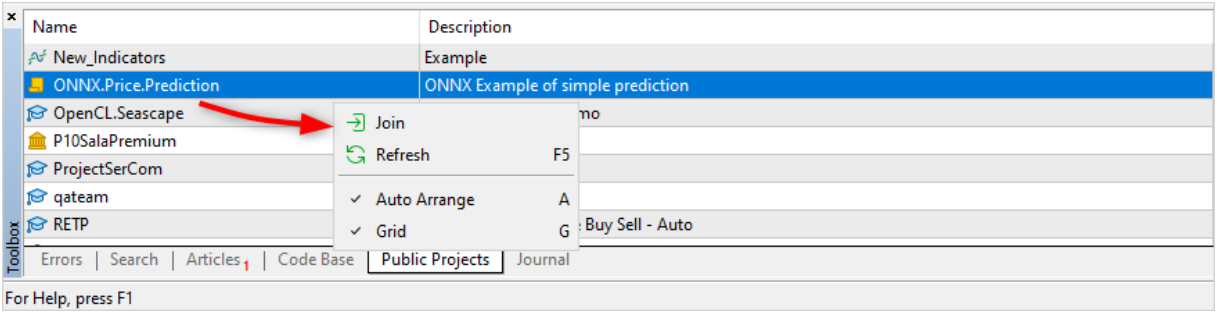
- copy_rates_from - ottiene le barre a partire dalla data specificata
- copy_rates_from_pos - ottiene le barre a partire dall'indice specificato
- copy_rates_range - ottiene le barre per l'intervallo di date specificato
- copy_ticks_from - ottiene i tick a partire dalla data specificata
- copy_ticks_range - ottiene i tick per l'intervallo di date specificato

Esempio di modello

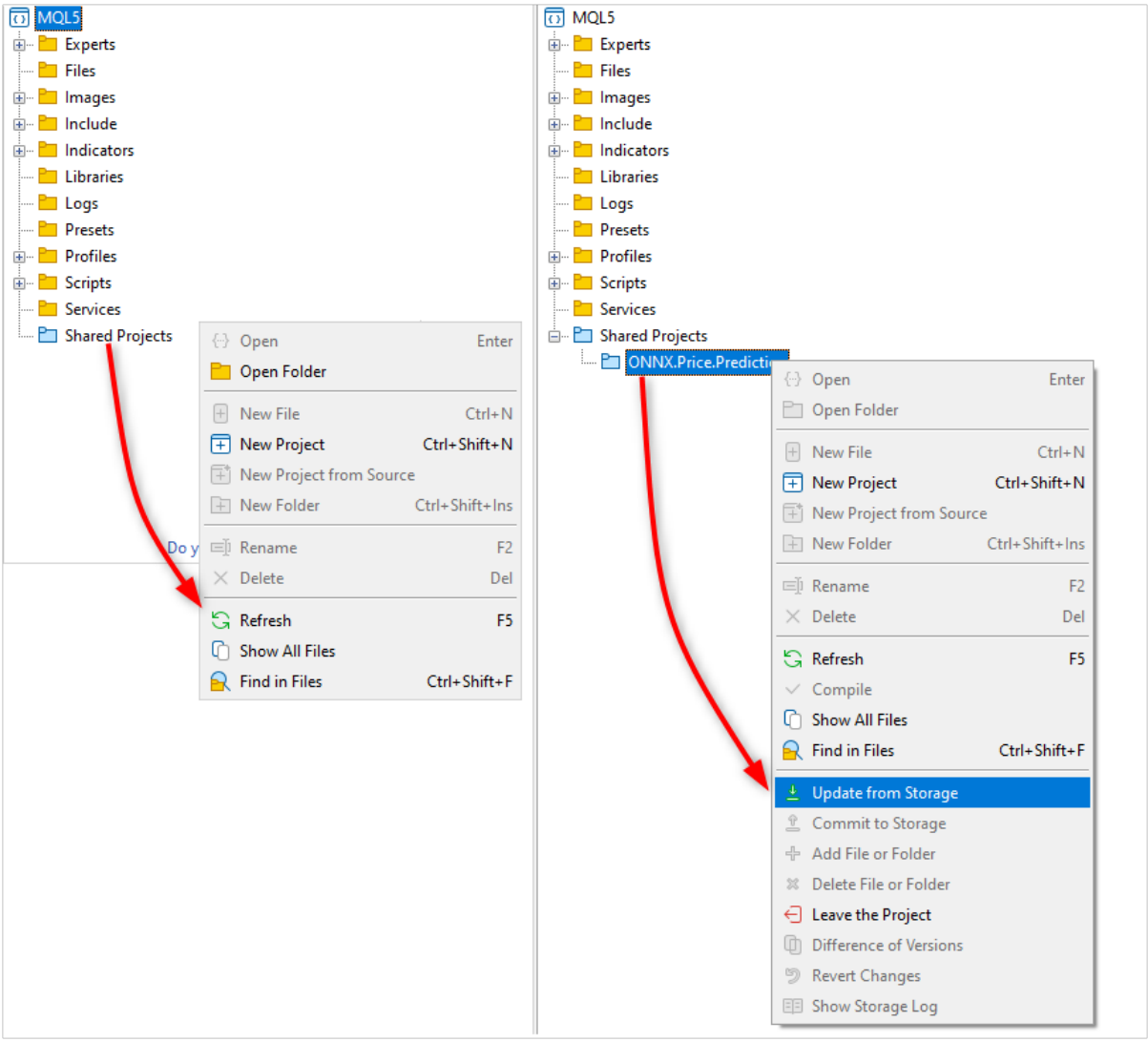
Un esempio di modello ONNX finito è disponibile in progetti pubblici. Si dovrebbe prima attivare MQL5 Storage nel Navigatore specificando il login MQL5 nelle impostazioni di Metaeditor (distinzione tra maiuscole e minuscole).



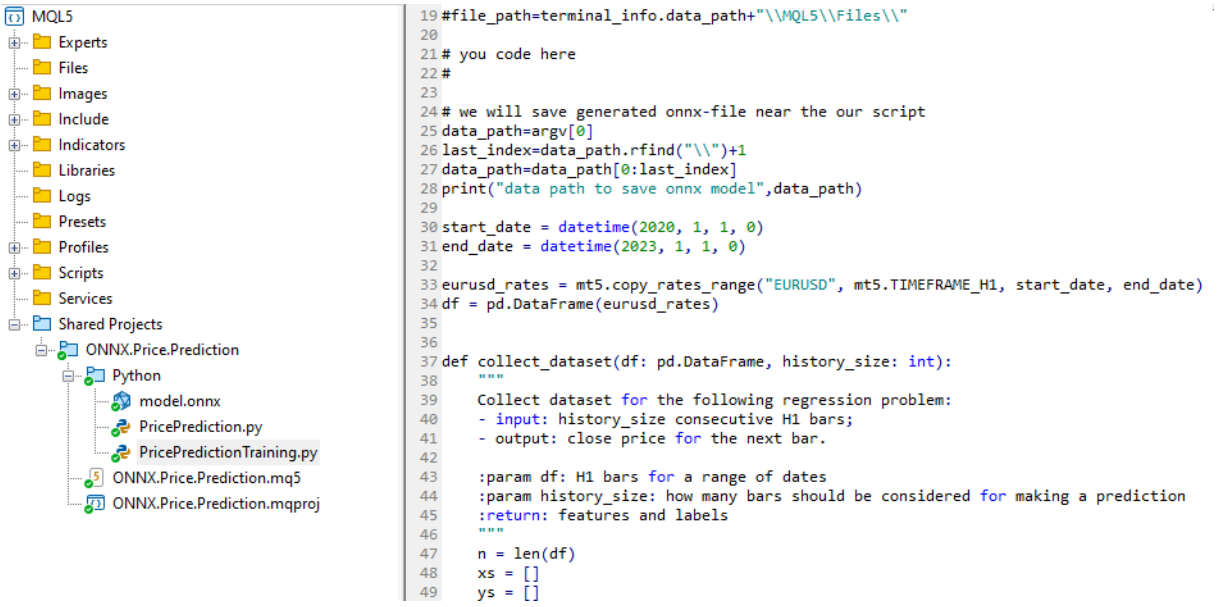
Dopo l'attivazione, trovare il progetto ONNX.Price.Prediction e unirlo tramite il comando del menu contestuale.



Successivamente, aggiornare il progetto da MQL5 Storage.



Il progetto contiene un modello ONNX, due script python, uno script MQL5 per le operazioni del progetto e un file di progetto MQL5 (ONNX.Price.Prediction.mqproj).

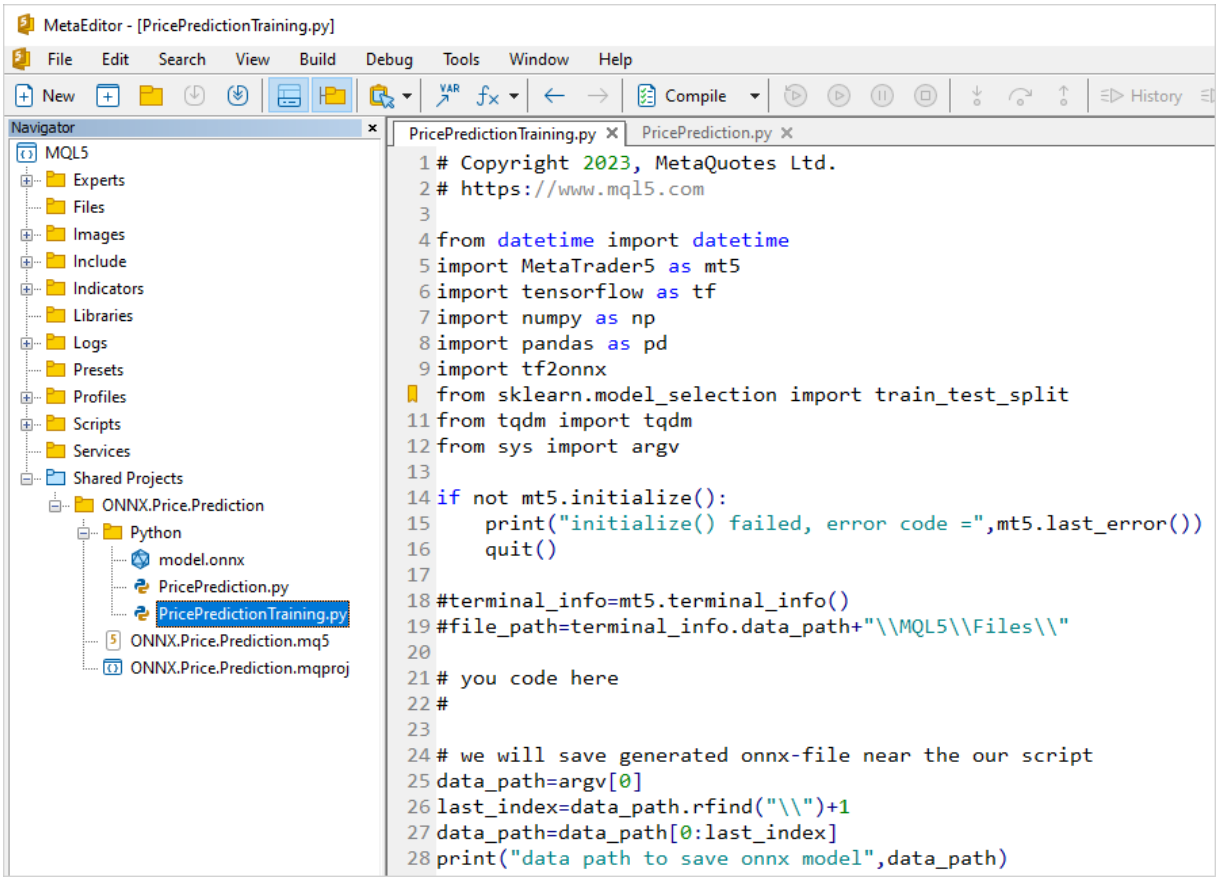



```
#file_path=terminal_info.data_path+"\\MQL5\\Files\\"

# you code here
#

# we will save generated onnx-file near the our script
data_path=argv[0]
last_index=data_path.rfind("\\")+1
data_path=data_path[0:last_index]
print("data path to save onnx model",data_path)

start_date = datetime(2020, 1, 1, 0)
end_date = datetime(2023, 1, 1, 0)

eurusd_rates = mt5.copy_rates_range("EURUSD", mt5.TIMEFRAME_H1, start_date, end_date)
df = pd.DataFrame(eurusd_rates)


def collect_dataset(df: pd.DataFrame, history_size: int):
    """
    Collect dataset for the following regression problem:
    - input: history_size consecutive H1 bars;
    - output: close price for the next bar.

    :param df: H1 bars for a range of dates
    :param history_size: how many bars should be considered for making a prediction
    :return: features and labels
    """
    n = len(df)
    xs = []
    ys = []
```

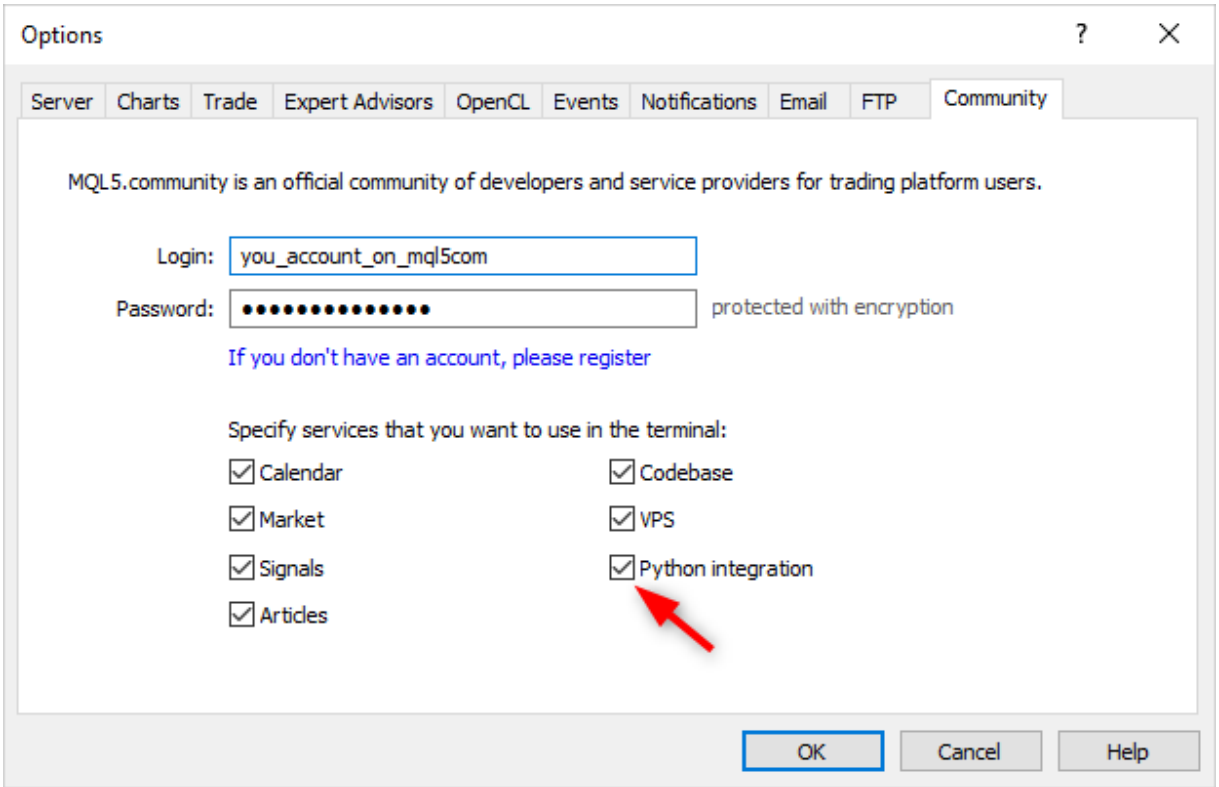
È possibile creare il modello ONNX da soli utilizzando lo script PricePredictionTraining.py incluso nel progetto. Per fare questo, è necessario prima installare i moduli richiesti dalla riga di comando.

```
python.exe -m pip install --upgrade pip
python -m pip install --upgrade tensorflow
python -m pip install --upgrade pandas
python -m pip install --upgrade scikit-learn
python -m pip install --upgrade matplotlib
python -m pip install --upgrade tqdm
python -m pip install --upgrade metatrader5
python -m pip install --upgrade onnx==1.12
python -m pip install --upgrade tf2onnx
python -m pip install --upgrade numpy
python -m pip install onnxruntime
```

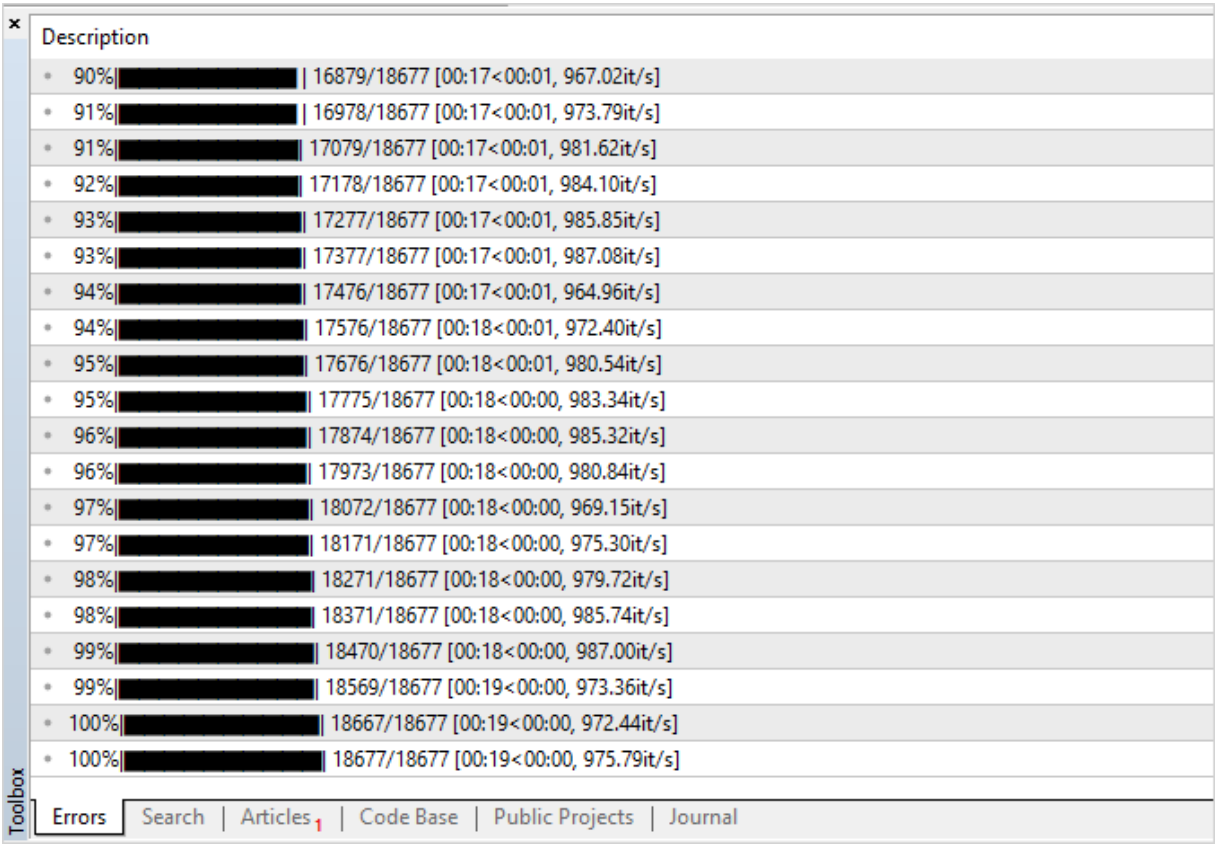
Dopo aver installato i moduli, aprire lo script PricePredictionTraining.py in Metaeditor ed eseguirlo con il pulsante Compila o con il tasto F7.



Prima di eseguire lo script Python, assicurarsi che il terminale MetaTrader 5 sia collegato ad un server con il simbolo EURUSD disponibile. Ad esempio, connettersi al server MetaQuotes-Demo e controllare "Integrazione con Python" nelle impostazioni del terminale.

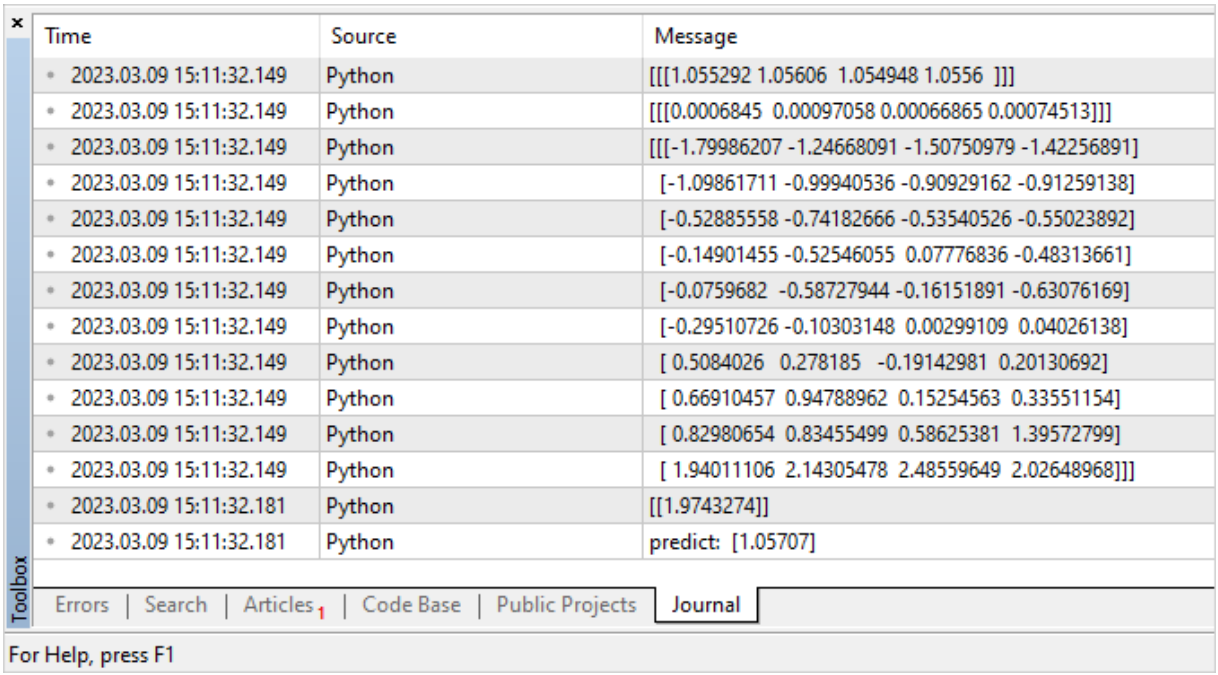


Durante l'addestramento della rete, Metaeditor stamperà i messaggi dallo script Python fino al completamento dell'addestramento.



Quando il risultato è 100%, il modello ONNX è pronto e viene salvato nella cartella del progetto all'indirizzo <cartella dati del terminale>\MQL5\Shared Projects\ONNX.Price.Prediction\Python.

È possibile controllare il modello risultante eseguendo il secondo script PricePrediction.py, premendo **F7**.



Esecuzione di un modello

Per eseguire un modello ONNX in MQL5, completare 3 passaggi:

1. Caricare il modello da un file *.onnx utilizzando la funzione OnnxCreate o da un array utilizzando OnnxCreateFromBuffer.
2. Specificare le dimensioni dei dati di ingresso e di uscita utilizzando le funzioni OnnxSetInputShape e OnnxSetOutputShape.
3. Eseguire il modello utilizzando la funzione OnnxRun, passando i parametri in ingresso e in uscita pertinenti.
4. Quando necessario, è possibile terminare l'operatività del modello utilizzando la funzione OnnxRelease.

Quando si crea un modello ONNX, si dovrebbero considerare i limiti e le restrizioni esistenti, descritti in https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs/OperatorKernels.md

Di seguito sono riportati alcuni esempi di tali restrizioni:

Operazione	Tipi di dati supportati
ReduceSum	tensor(double), tensor(float), tensor(int32), tensor(int64)
Mul	tensor(bfloat16), tensor(double), tensor(float), tensor(float16), tensor(int32), tensor(int64), tensor(uint32), tensor(uint64)

Di seguito è riportato un esempio di codice MQL5 dal progetto pubblico ONNX.Price.Prediction.

```
const long   ExtOutputShape[] = {1,1};     // dimensione del modello d'uscita
const long   ExtInputShape [] = {1,10,4}; // dimensione del modello d'ingresso
#resource "Python/model.onnx" as uchar ExtModel[]// modello come risorsa
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//-- ottenere 10 barre
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//---inserire un insieme di vettori OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//-- riempire le matrici di normalizzazione
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
```

```
      ms.Row(s,i);
     }
//-- normalizzare i dati d'ingresso
   x_norm-=mm;
   x_norm/=ms;
//-- creare il modello
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//-- specificare la dimensione dei dati d'ingresso
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- specificare la dimensione dei dati d'uscita
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- convertire i dati d'ingresso normalizzati in tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//-- ottenere i dati d'uscita del modello qui, cioè la previsione dei prezzi
   vectorf y_norm(1);
//-- eseguire il modello
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- stampa il valore di uscita del modello nel log
   Print(y_norm);
//-- fare la trasformazione inversa per ottenere il prezzo previsto
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//-- operazione completata
   OnnxRelease(handle);
   return(0);
  }
```

Esempio di esecuzione di uno script:

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
```

```
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
[0.28188983]
predicted 1.0559258806393044
```

Il terminale MetaTrader 5 ha selezionato l'esecutore ottimale per i calcoli — ONNX Runtime Execution Provider. In questo esempio, il modello è stato eseguito sulla CPU.

Modifichiamo lo script per calcolare la percentuale di successo delle previsioni dei prezzi di Chiusura effettuate in base ai valori delle precedenti 10 barre.

```
#resource "Python/model.onnx" as uchar ExtModel[]// modello come risorsa

#define TESTS 10000  // numero di set di dati del test
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
  {
//-- creare il modello
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Cannot create model. Error ",GetLastError());
      return(-1);
     }

//--- poiché la dimensione del tensore d'ingresso non è definita per il modello, speci
//--- il primo indice è la dimensione del lotto, il secondo indice è la dimensione del
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("OnnxSetInputShape error ",GetLastError());
      return(-2);
     }

//--- poiché la dimensione del tensore d'uscita non è definita per il modello, specifi
//--- il primo indice è la dimensione del lotto, deve coincidere alla dimensione del l
//--- il secondo indice è il numero dei prezzi previsti (solo la Chiusura è prevista c
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("OnnxSetOutputShape error ",GetLastError());
      return(-3);
```

```
     }
//--- escuzione dei test
   vector closes(TESTS);       // vettore per memorizzare i prezzi di convalida
   vector predicts(TESTS);     // vettore per memorizzare le previsioni ottenute
   vector prev_closes(TESTS); // vettore per memorizzare i prezzi precedenti

   matrix rates;               // matrice per ottenere la serie OHLC
   matrix splitted[2];         // due sottomatrici per dividere la serie in test e vali
   ulong  parts[]={10,1};      // dimensioni delle sottomatrici divise

//-- parte dalla barra precedente
   for(int i=1; i<=TESTS; i++)
     {
      //--- ottiene 11 barre
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
      //--- dividere la matrice in test e validazione
      rates.Vsplit(parts,splitted);
      //--- prendere il prezzo di Chiusura dalla matrice di convalida
      closes[i-1]=splitted[1][3][0];
      //---l'ultima Chiusura nella serie testata
      prev_closes[i-1]=splitted[0][3][9];

      //--- sottoporre al test la matrice di prova di 10 bar
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
      //--- errore di runtime
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//-- operazione completata
   OnnxRelease(session_handle);
//-- valutare se il movimento dei prezzi è stato previsto correttamente
   int    right_directions=0;
   vector delta_predicts=prev_closes-predicts;
   vector delta_actuals=prev_closes-closes;

   for(int i=0; i<TESTS; i++)
      if((delta_predicts[i]>0 && delta_actuals[i]>0) || (delta_predicts[i]<0 && delta_
         right_directions++;
   PrintFormat("right direction predictions = %.2f%%",(right_directions*100.0)/double
//---
   return(0);
  }
//+------------------------------------------------------------------+
// Preparare i dati ed eseguire il modello                           |
//+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
```

```
  {
   static matrixf input_data(10,4); // matrice per l'ingresso trasformato
   static vectorf output_data(1);   // vettore per ricevere il risultato
   static matrix mm(10,4);          // matrice di vettori orizzontali Mean</T11><segme
   static matrix ms(10,4);          // matrice di vettori orizzontali Std

//-- un insieme di vettori verticali OHLC deve essere inserito nel modello
   matrix x_norm=rates.Transpose();
//-- normalizzare i prezzi
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
   x_norm-=mm;
   x_norm/=ms;

//-- eseguire il modello
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("OnnxRun error ",GetLastError());
      return(0);
     }
//--- normalizzare il prezzo dal valore di uscita
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

Esegui lo script: l'accuratezza della previsione è di circa il 51%

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
right direction predictions = 51.34 %
```

Convalida del modello nello Strategy Tester

I modelli creati per le operazioni sui mercati finanziari possono essere convalidati nel terminale MetaTrader 5 Strategy Tester. Questa è l'opzione più veloce e conveniente, che elimina la necessità di emulare ulteriormente l'ambiente di mercato e le condizioni di trading.

Per testare il modello, creiamo un Expert Advisor basato sul codice del progetto pubblico ONNX.Price.Prediction. Questo richiederà alcune modifiche.

Spostarsi per la creazione del modello nella funzione OnInit. La sessione onnx sarà chiusa in OnDeinit. Individua il blocco delle operazioni del modello principale nel gestore OnTick.

Inoltre, aggiungere il prezzo della Chiusura ottenuto delle due barre precedenti, che è necessario per confrontare il prezzo attuale di Chiusura e la previsione.

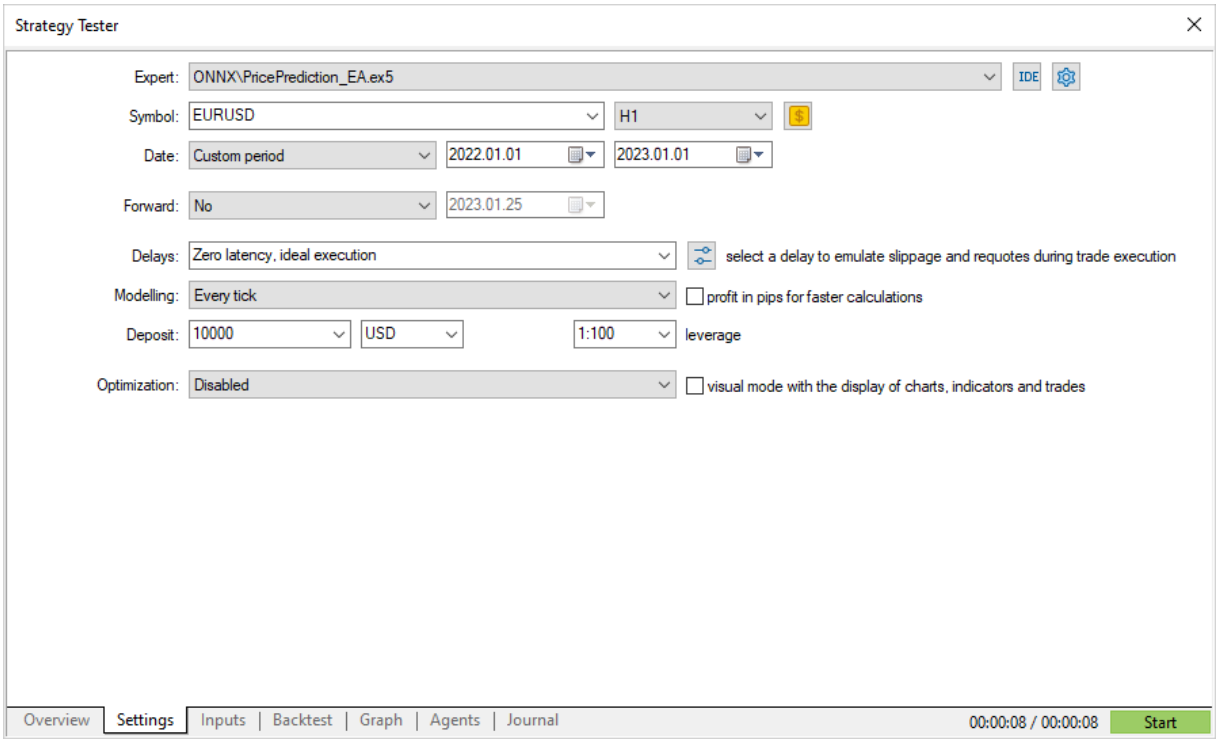
Il codice dell'Expert Advisor è piccolo e facile da leggere.

```
const long   ExtInputShape [] = {1,10,4}; // dimensione del modello d'ingresso
const long   ExtOutputShape[] = {1,1};    // dimensione del modello d'uscita
#resource "Python/model.onnx" as uchar ExtModel[];// modello come risorsa

long handle;          // handle del modello
ulong predictions=0; // contatore delle previsioni
ulong confirmed=0;   // contatore previsioni con successo
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//-- controlli di base
   if(_Symbol!="EURUSD")
     {
      Print("Symbol must be EURUSD, testing aborted");
      return(-1);
     }
   if(_Period!=PERIOD_H1)
     {
      Print("Timeframe must be H1, testing aborted");
      return(-1);
     }
//-- creare il modello
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//-- specificare la dimensione dei dati d'ingresso
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

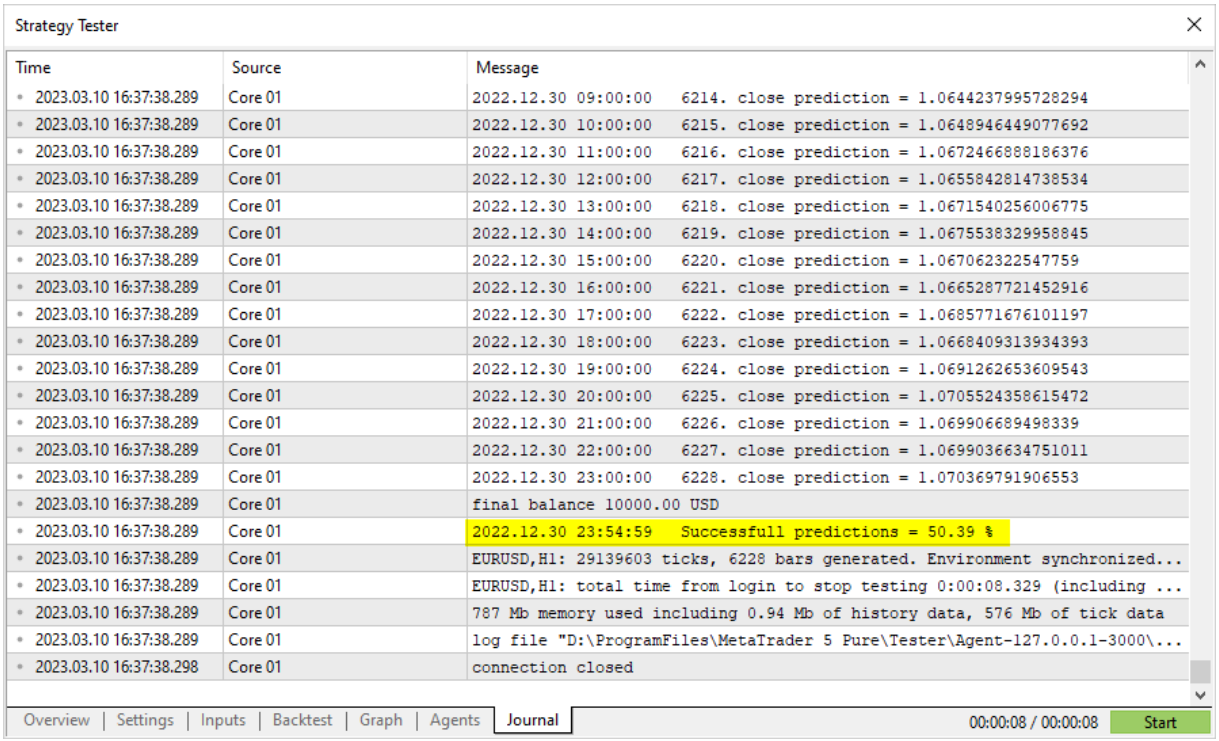
```
//-- specificare la dimensione dei dati d'uscita
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//-- operazione del modello completa
   OnnxRelease(handle);
//-- calcolo e statistiche di previsione d'uscita
   PrintFormat("Successfull predictions = %.2f %%",confirmed*100./double(predictions))
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//-- controlla l'orario di apertura della barra corrente
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Failed to get Time(0), error %d", GetLastError());
      return;
     }
//-- se l'orario di apertura non è cambiato, uscire fino alla prossima chiamata OnTic
   if(time==open_time)
      return;
//-- ottenere i prezzi di Chiusura delle ultime due barre completate
   double close[];
   int recieved=CopyClose(_Symbol,_Period,1,2,close);
   if(recieved!=2)
     {
      PrintFormat("CopyClose(2 bars) failed, error %d",GetLastError());
      return;
     }
   double delta_predict=predict-close[0]; // variazione dei prezzi prevista
   double delta_actual=close[1]-close[0]; // variazione effettiva dei prezzi
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;
```

```
//-- calcolare il prezzo di Chiusura sulla nuova barra per convalidare il prezzo sulla
   matrix rates;
//-- ottenere 10 barre
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//---inserire un insieme di vettori OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//-- riempire le matrici di normalizzazione
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//-- normalizzare i dati d'ingresso
   x_norm-=mm;
   x_norm/=ms;
//-- convertire i dati d'ingresso normalizzati in tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//-- ottenere i dati d'uscita del modello qui, cioè la previsione dei prezzi
   vectorf y_norm(1);
//-- eseguire il modello
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
     }
//-- fare la trasformazione inversa per ottenere il prezzo previsto e per convalidarlo
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // aumentare il contatore delle previsioni
   Print(predictions,". close prediction = ",predict);
//-- salva l'orario di apertura della barra per controllarlo al tick successivo
   open_time=time;
  }
```

Compilare l'Expert Advisor ed eseguire il test nel periodo dell'anno 2022. Specificare EURUSD con il timeframe H1, che sono i dati su cui il modello è stato addestrato. La modalità di modellazione tick può essere ignorata, poiché il codice controlla la nascita di una nuova barra.



Esegui e controlla il risultato in testing journal. Si nota che poco più del 50% delle previsioni erano corrette nel 2022.



Strategy Tester

Time	Source	Message
2023.03.10 16:37:38.289	Core 01	2022.12.30 09:00:00 6214. close prediction = 1.0644237995728294
2023.03.10 16:37:38.289	Core 01	2022.12.30 10:00:00 6215. close prediction = 1.0648946449077692
2023.03.10 16:37:38.289	Core 01	2022.12.30 11:00:00 6216. close prediction = 1.0672466888186376
2023.03.10 16:37:38.289	Core 01	2022.12.30 12:00:00 6217. close prediction = 1.0655842814738534
2023.03.10 16:37:38.289	Core 01	2022.12.30 13:00:00 6218. close prediction = 1.0671540256006775
2023.03.10 16:37:38.289	Core 01	2022.12.30 14:00:00 6219. close prediction = 1.0675538329958845
2023.03.10 16:37:38.289	Core 01	2022.12.30 15:00:00 6220. close prediction = 1.067062322547759
2023.03.10 16:37:38.289	Core 01	2022.12.30 16:00:00 6221. close prediction = 1.0665287721452916
2023.03.10 16:37:38.289	Core 01	2022.12.30 17:00:00 6222. close prediction = 1.0685771676101197
2023.03.10 16:37:38.289	Core 01	2022.12.30 18:00:00 6223. close prediction = 1.0668409313934393
2023.03.10 16:37:38.289	Core 01	2022.12.30 19:00:00 6224. close prediction = 1.0691262653609543
2023.03.10 16:37:38.289	Core 01	2022.12.30 20:00:00 6225. close prediction = 1.0705524358615472
2023.03.10 16:37:38.289	Core 01	2022.12.30 21:00:00 6226. close prediction = 1.069906689498339
2023.03.10 16:37:38.289	Core 01	2022.12.30 22:00:00 6227. close prediction = 1.0699036634751011
2023.03.10 16:37:38.289	Core 01	2022.12.30 23:00:00 6228. close prediction = 1.070369791906553
2023.03.10 16:37:38.289	Core 01	final balance 10000.00 USD
2023.03.10 16:37:38.289	Core 01	2022.12.30 23:54:59 Successfull predictions = 50.39 %
2023.03.10 16:37:38.289	Core 01	EURUSD,H1: 29139603 ticks, 6228 bars generated. Environment synchronized...
2023.03.10 16:37:38.289	Core 01	EURUSD,H1: total time from login to stop testing 0:00:08.329 (including ...
2023.03.10 16:37:38.289	Core 01	787 Mb memory used including 0.94 Mb of history data, 576 Mb of tick data
2023.03.10 16:37:38.289	Core 01	log file "D:\ProgramFiles\MetaTrader 5 Pure\Tester\Agent-127.0.0.1-3000\...
2023.03.10 16:37:38.298	Core 01	connection closed

Se il test preliminare del modello ha generato risultati soddisfacenti, è possibile iniziare a scrivere una strategia di trading a tutti gli effetti sulla base di questo modello.

OnnxCreate

Crea una sessione ONNX, caricando un modello da un file *.onnx

```
long  OnnxCreate(
   string  filename,  // percorso del file
   uint    flags      // flag per creare il modello
   );
```

Parametri

filename

[in] Percorso del file *.onnx del modello relativo alla cartella \MQL5\Files\ .

flags

[in] Flag da ENUM_ONNX_FLAGS, che descrive la modalità di creazione del modello: ONNX_COMMON_FOLDER e ONNX_DEBUG_LOGS.

Valore Restituito

L'handle della sessione creata o INVALID_HANDLE se si verifica un errore. Per ottenere il codice errore, chiamare la funzione GetLastError.

Nota

Se il file specificato non viene trovato sul disco, il sistema riprova ad aprire il file, aggiungendo l'estensione '.onnx' al nome.

OnnxCreateFromBuffer

Crea una sessione ONNX, caricando un modello da un array di dati

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],   // array di riferimento
   ulong         flags       // flagper la creazione del modello
   );
```

Parametri

buffer

[in] Array con dati del modello ONNX.

flags

[in] Flag da ENUM_ONNX_FLAGS, che descrive la modalità di creazione del modello: ONNX_COMMON_FOLDER e ONNX_DEBUG_LOGS.

Valore Restituito

L'handle della sessione creata o INVALID_HANDLE se si verifica un errore. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxRelease

Chiude una sessione ONNX

```
bool  OnnxRelease(
   long   onnx_handle  // handle sessione ONNX
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

Valore Restituito

Restituisce true in caso di successo; altrimenti restituisce false. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxRun

Esegue un modello ONNX.

```
bool  OnnxRun(
 long   onnx_handle,  // handle sessione ONNX
 ulong  flags,        // flag che descrive la modalità di esecuzione
   ...                // ingressi e uscite del modello
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

flags

[in] Flag da ENUM_ONNX_FLAGS che descrive la modalità di esecuzione: ONNX_DEBUG_LOGS e ONNX_NO_CONVERSION.

...

[in] [out] Ingressi e uscite del modello.

Restituisce true in caso di successo o false altrimenti. Per ottenere il codice errore, chiamare la funzione GetLastError.

ENUM_ONNX_FLAGS

ID	Descrizione
ONNX_DEBUG_LOGS	Log di debug di output
ONNX_NO_CONVERSION	Disattiva la conversione automatica, utilizza i dati dell'utente come sono
ONNX_COMMON_FOLDER	Carica un file modello dalla cartella Common\Files; il valore è uguale a FILE_COMMON flag

Esempio:

```
const long                              ExtOutputShape[] = {1,1};     // dimensione del
const long                              ExtInputShape [] = {1,10,4}; // forma del model
#resource "Python/model.onnx" as uchar ExtModel[]                    // modello come r
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//-- ottenere 10 barre
```

```
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//---inserire un insieme di vettori OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//-- riempire le matrici di normalizzazione
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//-- normalizzare i dati d'ingresso
   x_norm-=mm;
   x_norm/=ms;
//-- creare il modello
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//-- specificare la dimensione dei dati d'ingresso
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- specificare la dimensione dei dati d'uscita
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- convertire i dati d'ingresso normalizzati in tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//-- ottenere i dati d'uscita del modello qui, cioè la previsione dei prezzi
   vectorf y_norm(1);
//-- eseguire il modello
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- stampa il valore di uscita del modello nel log
   Print(y_norm);
//-- fare la trasformazione inversa per ottenere il prezzo previsto
   double y_pred=y_norm[0]*s[3]+m[3];
```

```
   Print("price predicted:",y_pred);
//-- operazione completata
   OnnxRelease(handle);
   return(0);
  };
```

Vedi anche

OnnxSetInputShape, OnnxSetOutputShape

OnnxGetInputCount

Ottiene il numero di ingressi in un modello ONNX.

```
long  OnnxGetInputCount(
   long   onnx_handle  // handle sessione ONNX
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

Valore Restituito

Restituisce il numero di parametri in ingresso in caso di successo; altrimenti restituisce -1. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxGetOutputCount

Ottiene il numero di uscite in un modello ONNX.

```
long  OnnxGetOutputCount(
   long   onnx_handle  // handle sessione ONNX
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

Valore Restituito

Restituisce il numero di parametri d'uscita in caso di successo; altrimenti restituisce -1. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxGetInputName

Ottiene il nome dell'ingresso di un modello tramite indice

```
string  OnnxGetInputName(
   long   onnx_handle,  // handle sessione ONNX
   long   index         // indice parametro
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

index

[in] Indice del parametro d'ingresso, partendo da 0.

Valore Restituito

Restituisce il nome del parametro d'ingresso in caso di successo; altrimenti restituisce NULL. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxGetOutputName

Ottiene il nome dell'uscita di un modello tramite indice.

```
string  OnnxGetOutputName(
   long   onnx_handle,  // handle sessione ONNX
   long   index         // indice parametro
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

index

[in] Indice del parametro d'uscita, partendo da 0.

Valore Restituito

Restituisce il nome del parametro d'uscita in caso di successo; altrimenti restituisce NULL. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxGetInputTypeInfo

Ottiene la descrizione del tipo di ingresso dal modello.

```
bool  OnnxGetInputTypeInfo(
   long           onnx_handle,  // handle sessione ONNX
   long           index,        // indice parametro
   OnnxTypeInfo&  typeinfo      // descrizione del tipo di parametro
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

index

[in] Indice del parametro d'ingresso, partendo da 0.

typeinfo

[out] La struttura OnnxTypeInfo descrive il tipo del parametro d'ingresso.

Valore Restituito

Restituisce true in caso di successo; altrimenti restituisce false. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxGetOutputTypeInfo

Ottiene la descrizione del tipo di uscita dal modello.

```
bool  OnnxGetOutputTypeInfo(
   long           onnx_handle,  // handle sessione ONNX
   long           index,        // indice parametro
   OnnxTypeInfo&  typeinfo      // descrizione del tipo di parametro
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

index

[in] Indice del parametro d'uscita, partendo da 0.

typeinfo

[out] La struttura OnnxTypeInfo che descrive il tipo del parametro d'uscita.

Valore Restituito

Restituisce true in caso di successo; altrimenti restituisce false. Per ottenere il codice errore, chiamare la funzione GetLastError.

OnnxSetInputShape

Imposta la dimensione dei dati d'ingresso di un modello tramite indice.

```
bool  OnnxSetInputShape(
   long           onnx_handle,  // handle sessione ONNX
   long           input_index,  // indice parametro d'ingresso
   const ulong&   shape[]       // array che descrive la dimensione dei dati d'ingresso
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

input_index

[in] Indice del parametro d'ingresso, partendo da 0.

shape

[in] Array che descrive la dimensione dei dati d'ingresso del modello.

Valore Restituito

Restituisce il nome del parametro d'ingresso in caso di successo; altrimenti restituisce NULL. Per ottenere il codice errore, chiamare la funzione GetLastError.

Esempio:

```
//---- descrivere le dimensioni dei dati d'ingresso e d'uscita del modello
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//-- creare il modello
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//-- specificare la dimensione dei dati d'ingresso
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- specificare la dimensione dei dati d'uscita
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Vedi anche

OnnxSetOutputShape

OnnxSetOutputShape

Imposta la dimensione dei dati di uscita di un modello tramite indice.

```
bool  OnnxSetOutputShape(
   long           onnx_handle,    // handle sessione ONNX
   long           output_index,   // indice parametro d'uscita
   const ulong&   shape[]         // array che descrive la dimensione dei dati d'uscita
   );
```

Parametri

onnx_handle

[in] Handle ONNX dell'oggetto della sessione creato tramite OnnxCreate o OnnxCreateFromBuffer.

output_index

[in] Indice del parametro d'uscita, partendo da 0.

shape

[in] Array che descrive la dimensione dei dati d'uscita del modello.

Valore Restituito

Restituisce il nome del parametro d'ingresso in caso di successo; altrimenti restituisce NULL. Per ottenere il codice errore, chiamare la funzione GetLastError.

Esempio:

```
//---- descrivere le dimensioni dei dati d'ingresso e d'uscita del modello
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//-- creare il modello
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//-- specificare la dimensione dei dati d'ingresso
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//-- specificare la dimensione dei dati d'uscita
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Vedi anche

OnnxSetInputShape

Strutture dati

Le seguenti strutture dati sono utilizzate per operazioni con modelli ONNX:

OnnxTypeInfo

La struttura descrive il tipo di un ingresso o parametro d'uscita di un modello ONNX

```
struct OnnxTypeInfo
 {
  ENUM_ONNX_TYPE         type;          // tipo di parametro
  OnnxTensorTypeInfo     tensor;        // descrizione del tensore
  OnnxMapTypeInfo        map;           // descrizione della mappa
  OnnxSequenceTypeInfo   sequence;      // descrizione della sequenza
 };
```

Solo il tensore (ONNX_TYPE_TENSOR) può essere usato come ingresso. In questo caso, solo il campo OnnxTypeInfo::tensor è riempito con valori, mentre gli altri campi (mappa e sequenza) non sono definiti.

Solo uno dei tre tipi OnnxTypeInfo (ONNX_TYPE_TENSOR, ONNX_TYPE_MAP o ONNX_TYPE_SEQUENCE) può essere utilizzato come ingresso. La sottostruttura corrispondente (OnnxTypeInfo::tensor, OnnxTypeInfo::map o OnnxTypeInfo::sequence) viene riempita a seconda del tipo.

OnnxTensorTypeInfo

La struttura descrive il tensore in ingresso o parametro d'uscita di un modello ONNX

```
struct OnnxTensorTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE  data_type;      // tipo di dati nel tensore
  const long                 dimensions[];   // numero di elementi nel tensore
 };
```

OnnxMapTypeInfo

La struttura descrive la mappa ottenuta nel parametro d'uscita di un modello ONNX

```
struct OnnxMapTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE  key_type;      // tipo di chiave
  const OnnxTypeInfo&        value_type;    // tipo di valore
 };
```

OnnxSequenceTypeInfo

La struttura descrive la sequenza ottenuta nel parametro d'uscita di un modello ONNX

```
struct OnnxSequenceTypeInfo
```

```
{
 const OnnxTypeInfo&        value_type;      // tipo di dati nella sequenza
};
```

ENUM_ONNX_TYPE

L'enumerazione **ENUM_ONNX_TYPE** definisce il tipo di parametro di un modello

ID	Descrizione
ONNX_TYPE_UNKNOWN	Sconosciuto
ONNX_TYPE_TENSOR	Tensore
ONNX_TYPE_SEQUENCE	Sequenza
ONNX_TYPE_MAP	Mappa
ONNX_TYPE_OPAQUE	Astratto (opaco)
ONNX_TYPE_SPARSETENSOR	Tensore sparso

ENUM_ONNX_DATA_TYPE

L'enumerazione >**ENUM_ONNX_DATA_TYPE** definisce il tipo di dati usato

ID	Descrizione
ONNX_DATA_TYPE_UNDEFINED	Non definito
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	int 8-bit
ONNX_DATA_TYPE_UINT16	uint 16-bit
ONNX_DATA_TYPE_INT16	int 16-bit
ONNX_DATA_TYPE_INT32	int 32-bit
ONNX_DATA_TYPE_INT64	int 64-bit
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	float 16-bit
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint 32-bit
ONNX_DATA_TYPE_UINT64	uint 64-bit

ID	Descrizione
ONNX_DATA_TYPE_COMPLEX64	numeri complessi a 64-bit
ONNX_DATA_TYPE_COMPLEX128	numeri complessi a 128-bit
ONNX_DATA_TYPE_BFLOAT16	bfloat 16-bit (Brain Floating Point)

ENUM_ONNX_FLAGS

L'enumerazione **ENUM_ONNX_FLAGS** definisce la modalità di esecuzione del modello

ID	Descrizione
ONNX_DEBUG_LOGS	Log di debug di output
ONNX_NO_CONVERSION	Disattiva la conversione automatica, utilizza i dati dell'utente come sono
ONNX_COMMON_FOLDER	Carica un file modello dalla cartella Common\Files; il valore è uguale a FILE_COMMON flag

Libreria Standard

Questo gruppo di capitoli contiene i dettagli tecnici della libreria standard MQL5 e le descrizioni di tutte le sue componenti principali.

La MQL5 Standard Library è scritta in MQL5 ed è progettata per facilitare la scrittura di programmi (indicatori, script, experts) per gli utenti finali. LA libreria offre un comodo accesso alla maggior parte delle funzioni MQL5 interne.

La MQL5 Standard Library si trova nella directory di lavoro del terminale nella cartella 'Include'.

Sezione	Locazione
Mathematics	Include\Math\
OpenCL	Include\OpenCL\
Basic Class CObject	Include\
Collezione Dati	Include\Arrays\
Collezioni Dati Generali	Include\Generic\
File	Include\Files\
Stringhe	Include\Strings\
Oggetti Grafici	Include\Objects\
Grafici Personalizzati	Include\Canvas\
Grafici Scientifici	Include\Canvas\
Chart Prezzi	Include\Charts\
Grafici Scientifici	Include\Graphics\
Indicatori	Include\Indicators\
Classi di Trade	Include\Trade\
Moduli Strategia	Include\Expert\
Pannelli e Dialoghi	Include\Controls\

Matematiche

Librerie multiple vengono fornite per eseguire calcoli in diverse aree delle matematiche:

- Statistiche – funzioni per lavorare con varie distribuzioni di teoria della Probabilità
- Logica Fuzzy – libreria che implementa sistemi di inferenza fuzzy Mamdani e Sugeno
- ALGLIB – analisi dei dati (clustering, alberi decisionali, regressione lineare, reti neurali), soluzione di equazioni differenziali, trasformata di Fourier, integrazione numerica, problemi di ottimizzazione, analisi statistiche, e altro ancora.

Statistiche

La libreria statistica fornisce un modo conveniente di lavorare con distribuzioni statistiche di base.

La libreria fornisce 5 funzioni per ogni distribuzione:

1. Calcolo della densità di probabilità - funzioni della forma MathProbabilityDensityX()
2. Calcolo delle probabilità - funzioni della forma MathCumulativeDistributionX()
3. Calcolo dei quantili di distribuzione - funzioni della forma MathQuantileX()
4. Generazione di numeri casuali con distribuzione specificata - funzioni della forma MathRandomX()
5. Calcolo dei momenti teorici delle distribuzioni - funzioni della forma MathMomentsX()

Oltre a calcolo dei valori per le singole variabili casuali, la libreria fornisce anche overloads per le funzioni che eseguono gli stessi calcoli per gli array.

- Statistical Characteristics
- Distribuzione normale
- Distribuzione log-normale
- Distribuzione Beta
- Distribuzione beta non centrale
- Distribuzione gamma
- Distribuzione del chi quadrato
- Distribuzione chi-quadrato non centrale
- Distribuzione esponenziale
- Distribuzione-F
- Distribuzione-F non centrale
- Distribuzione-t
- Distribuzione-T non centrale
- Distribuzione logistica
- Distribuzione di Cauchy
- Distribuzione uniforme
- Distribuzione di Weibull
- Distribuzione binomiale
- Distribuzione binomiale negativa
- Distribuzione geometrica
- Distribuzione ipergeometrica
- Distribuzione di Poisson
- Subfunzioni

Esempio:

```
//+------------------------------------------------------------------+
//|                                     NormalDistributionExample.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- include le funzioni per calcolare la distribuzione normale
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| Funzione start programma Script function                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- impostai parametri della distribuzione normale
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("Distribuzione normale con i parametri mu=%G e sigma=%G, esempi di calc
//--- imposta l'intervallo
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- variabili per il calcolo della probabilità
   double cdf1,cdf2,probability;
//--- variabili per i codici errore
   int error_code1,error_code2;
//--- calcola i valori della funzione di distribuzione
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- controlla i codici errore
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- calcola la probabilità di una variabile random nel range
      probability=cdf2-cdf1;
      //--- da in output il risultato
      PrintFormat("1. Calcola la probabilità di una variabile random entro il range di
      PrintFormat("  Risposta: Probabilità = %5.8f",probability);
     }

//--- Trova il valore del range della variabile random x, corrispondente al 95% del li
   probability=0.95; //  imposta la confidenza di probabilità
//--- imposta le probabilità ai confini degli intervalli
   double p1=(1.0-probability)*0.5;
   double p2=probability+(1.0-probability)*0.5;
//--- calcola i confini degli intevalli
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- controlla codici errore
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- output del risultato
      PrintFormat("2. Per l'intervallo di confidenza = %.2f, trova il range della vari
      PrintFormat("  Risposta: il range è %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Calcola i primi 4 momenti calcolati e teorici della distribuzione")
//--- Genea un arrai di numeri random, calcola i primi 4 momenti e li confronta con i
   int data_count=1000000;  // imposta il numero di valori e prepara un array
   double data[];
   ArrayResize(data,data_count);
//--- genera i valori random e li memorizza nell'array
```

```
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- imposta l'indice dei valori iniziali e l'ammontare dei dati del calcolo
   int start=0;
   int count=data_count;
//--- calcola i primi 4 momenti dei valori generati
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- variabili per i momenti teorici
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- mostra i valori dei momenti calcolati
   PrintFormat("            Media           Varianza          Asimmetria           Cu
   PrintFormat("Calccolati  %.10f   %.10f      %.10f      %.10f",mean,variance,skewnes
//--- calcola i valori teorici dei momenti e li confronta con i valori ottenuti
   if(MathMomentsNormal(mu,sigma,normal_mean,normal_variance,normal_skewness,normal_ku
     {
      PrintFormat("Teorici %.10f   %.10f      %.10f       %.10f",normal_mean,normal_va
      PrintFormat("Differenza  %.10f   %.10f      %.10f       %.10f",mean-normal_mean,
     }
  }
```

Statistical Characteristics

Questo gruppo di funzioni calcola le caratteristiche statistiche del elementi dell'array:

- media,
- varianza,
- asimmetria,
- curtosi,
- mediana,
- radice-quadratica-media e
- deviazione standard.

Funzione	Descrizione
MathMean	Calcola la media (primo momento) di elementi dell'array
MathVariance	Calcola la varianza (secondo momento) di elementi dell'array
MathSkewness	Calcola l'asimmetria (terzo momento) di elementi dell'array
MathKurtosis	Calcola la curtosi (quarto momento) di elementi dell'array
MathMoments	Calcola i primi 4 momenti (media, varianza, asimmetria, curtosi) degli elementi dell'array
MathMedian	Calcola il valore mediano di elementi dell'array
MathStandardDeviation	Calcola la deviazione standard di elementi dell'array
MathAverageDeviation	Calcola la deviazione media assoluta di elementi dell'array

MathMean

Calcola la media (primo momento) degli elementi dell'array. Analogo di mean() in R.

```
double  MathMean(
   const double&  array[]                // array con dati
   );
```

Parametri

array

[in] Array con i dati per il calcolo della media.

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

La media di elementi dell'array. In caso di errore restituisce NaN (Not a Number (non un numero)).

MathVariance

Calcola la varianza (secondo momento) degli elementi dell'array. Analogo di var() in R.

```
double  MathVariance(
   const double&  array[]                // array con dati
   );
```

Parametri

array

[in] Array con i dati per il calcolo.

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

Varianza degli elementi di un array. In caso di errore restituisce NaN (Not a Number (non un numero)).

MathSkewness

Calcola l'asimmetria (terzo momento) degli elementi dell'array. Analogo di skrewness() in R (biblioteca e1071).

```
double  MathSkewness(
   const double&  array[]                // array con dati
   );
```

Parametri

array

[in] Array con i dati per il calcolo.

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

Asimmetria degli elementi di un array. In caso di errore restituisce NaN (Not a Number (non un numero)).

MathKurtosis

Calcola la curtosi (quarto momento) degli elementi dell'array. Analogo di kurtosis() in R (libreria e1071).

```
double  MathKurtosis(
   const double&  array[]                // array con dati
   );
```

Parametri

array

[in] Array con i dati per il calcolo.

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

Curtosi degli elementi di un array. In caso di errore restituisce NaN (Not a Number (non un numero)).

Disclaimer

Il calcolo della curtosi viene eseguito utilizzando l'eccesso di curtosi attorno alla distribuzione normale (curtosi d'eccesso=kurtosis-3), cioè l'eccesso di curtosi di una distribuzione normale è zero.

È positiva se il picco della distribuzione attorno al valore atteso è 'tagliente', e negativo se il picco è 'piatto'.

MathMoments

Calcola i primi 4 momenti (media, varianza, asimmetria, curtosi) degli elementi di un array.

```
double  MathMoments(
   const double&  array[],              // array con dati
   double&        mean,                 // media (1mo momento)
   double&        variance,             // varianza (2ndo momento)
   double&        skewness,             // asimmetria (3zo momento)
   double&        kurtosis,             // curtosi (4to momento)
   const int      start=0,              // indice iniziale
   const int      count=WHOLE_ARRAY     // il numero degli elementi
   );
```

Parametri

array

[in] Array con i dati per il calcolo.

mean

[out] Variabile per la media (1mo momento)

variance

[out] Variabile per la varianza (2ndo momento)

skewness

[out] Variabile per l'asimmetria (3zo momento)

kurtosis

[out] Variabile per la curtosi (4to momento)

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

Restituisce true se i momenti sono stati calcolati con successo, altrimenti false.

Disclaimer

Il calcolo della curtosi viene eseguito utilizzando l'eccesso di curtosi attorno alla distribuzione normale (curtosi d'eccesso=kurtosis-3), cioè l'eccesso di curtosi di una distribuzione normale è zero.

È positiva se il picco della distribuzione attorno al valore atteso è 'tagliente', e negativo se il picco è 'piatto'.

MathMedian

Calcola il valore mediano di elementi dell'array. Analogo di median() in R.

```
double  MathMedian(
   const double&  array[]                // array con dati
   );
```

Parametri

array

[in] Array con i dati per il calcolo.

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

Il valore mediano di elementi dell'array. In caso di errore restituisce NaN (Not a Number (non un numero)).

MathStandardDeviation

Calcola la deviazione standard degli elementi di un array. Analogo di sd() in R.

```
double  MathStandardDeviation(
   const double&  array[]                // array con dati
   );
```

Parametri

array

[in] Array con i dati per il calcolo.

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

La deviazione standard degli elementi di un array. In caso di errore restituisce NaN (Not a Number (non un numero)).

MathAverageDeviation

Calcola la deviazione media assoluta degli elementi di un array. Analogo di aad() in R.

```
double  MathAverageDeviation(
   const double&  array[]                // array con dati
   );
```

Parametri

array

[in] Array con i dati per il calcolo.

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

La deviazione media assoluta degli elementi di un array. In caso di errore restituisce NaN (Not a Number (non un numero)).

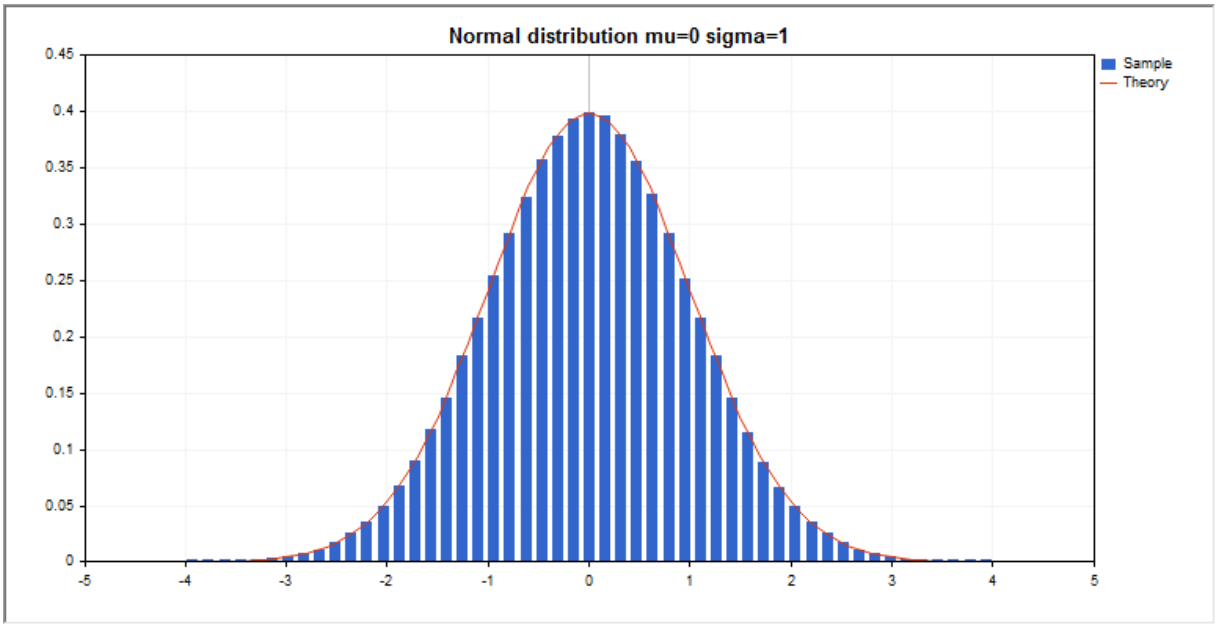
Distribuzione normale

Questa sezione contiene funzioni per lavorare con distribuzione normale. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge normale. La distribuzione è definita dalla seguente formula:

$$f_{Normal}(x|\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

dove:

- x — valore della variabile casuale
- μ — valore atteso
- σ — deviazione radice-quadratica-media



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityNormal	Calcola la funzione di densità di probabilità della distribuzione normale
MathCumulativeDistributionNormal	Calcola il valore della normale funzione di distribuzione di probabilità
MathQuantileNormal	Calcola il valore della funzione di distribuzione normale inversa per la probabilità specificata
MathRandomNormal	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge normale

Funzione	Descrizione
MathMomentsNormal	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione normale

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Normal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double mean_value=0;  // valore atteso (media)
input double std_dev=1;     // deviazione radice-quadratica-media (deviazione standard
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // il numero di valori nell'esempio
   int ncells=51;       // il numero di intervalli nell'istogramma
   double x[];          // centro degli intervalli dell'istogramma
   double y[];          // il numero di valori dall'esempio che cade all'interno dell
   double data[];       // esempio di valori casuali
   double max,min;      // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione normale
   MathRandomNormal(mean_value,std_dev,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNormal(x2,mean_value,std_dev,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Normal distribution mu=%G sigma=%G",mean_value
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
//--- disegna tutte le curve
   graphic.CurvePlotAll();
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

```
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNormal

Calcola il valore della funzione di densità di probabilità di distribuzione normale con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // parametro della media della distribuzione(valore a
   const double  sigma,          // parametro sigma della distribuzione (deviazione ra
   const bool    log_mode,       // calcola il logaritmo del valore
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione normale con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // parametro della media della distribuzione(valore a
   const double  sigma,          // parametro sigma della distribuzione (deviazione ra
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione normale con i parametri mu e sigma per una serie di variabili casuali x[]. In caso di errore restituisce false. Analogo didnorm() in R.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],            // array con i valori della variabile random
   const double   mu,             // parametro della media della distribuzione (valore
   const double   sigma,          // paramtero sigma della distribuzione (deviazione
   const bool     log_mode,       // calcola il logaritmo del valore
   double&        result[]        // array per i valori della funzione di densità di p
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione normale con i parametri mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],            // array con i valori della variabile random
   const double   mu,             // parametro della media della distribuzione (valore
   const double   sigma,          // paramtero sigma della distribuzione (deviazione
   double&        result[]        // array per i valori della funzione di densità di p
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] parametro della media della distribuzione (valore atteso).

sigma

[in] parametro sigma della distribuzione (deviazione radice-quadrata-media).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array per ottenere i valori della funzione di densità di probabilità.

MathCumulativeDistributionNormal

Calcola il valore della funzione di distribuzione normale con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   const bool    tail,           // flag per il calcolo della coda
   const bool    log_mode,       // calcola il logaritmo del valore
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione normale con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione normale con i parametri mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pnorm() in R.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   const bool    tail,           // flag per il calcolo della coda
   const bool    log_mode,       // calcola il logaritmo del valore
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione normale con i parametri mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] parametro della media della distribuzione (valore atteso).

sigma

[in] parametro sigma della distribuzione (deviazione radice-quadrata-media).

tail

[in] Flag di calcolo. se tail=true, allora viene calcolata la probabilità della variabile random x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array per ottenere i valori della funzione di probabilità.

MathQuantileNormal

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione normale inversa con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileNormal(
   const double  probability,    // valore probabilità della variabile random
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   const bool    tail,           // flag per il calcolo della coda
   const bool    log_mode,       // calcola il logaritmo del valore
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione normale inversa con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileNormal(
   const double  probability,    // valore probabilità della variabile random
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[ ]*di valori di probabilità, la funzione calcola i valori della funzione di distribuzione normale inversa con i parametri mu e sigma. In caso di errore restituisce false. Analogo di qnorm() in R.

```
bool  MathQuantileNormal(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   const bool    tail,           // flag per il calcolo della coda
   const bool    log_mode,       // calcola il logaritmo del valore
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[ ]*di valori di probabilità, la funzione calcola i valori della funzione di distribuzione normale inversa con i parametri mu e sigma. In caso di errore restituisce false.

```
bool  MathQuantileNormal(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  mu,             // valore atteso
   const double  sigma,          // deviazione radice-quadrata-media
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

mu

[in] parametro della media della distribuzione (valore atteso).

sigma

[in] parametro sigma della distribuzione (deviazione radice-quadrata-media).

tail

[in] Flag di calcolo. Se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] array per ottenere i quantili.

MathRandomNormal

Genera una variabile pseudocasuale distribuita secondo la legge normale con i parametri di mu e sigma. In caso di errore restituisce NaN.

```
double  MathRandomNormal(
   const double  mu,              // valore atteso
   const double  sigma,           // deviazione radice-quadrata-media
   int&          error_code       // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge normale con i parametri di mu e sigma. In caso di errore restituisce false. Analogo di rnorm() in R.

```
bool  MathRandomNormal(
   const double  mu,              // valore atteso
   const double  sigma,           // deviazione radice-quadrata-media
   const int     data_count,      // ammontare dei dati richiesti
   double&       result[]         // array per ottenere le variabili pseudocasuali
   );
```

Parametri

mu

[in] parametro della media della distribuzione (valore atteso).

sigma

[in] parametro sigma della distribuzione (deviazione radice-quadrata-media).

data_count

[in] Il numero di variabili pseudocasuali da ottenere.

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsNormal

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione beta

```
double  MathMomentsNormal(
   const double  mu,              // valore atteso
   const double  sigma,           // deviazione radice-quadrata-media
   double&       mean,            // variabile per la media
   double&       variance,        // variabile per la varianza
   double&       skewness,        // variabile per l'asimmetria
   double&       kurtosis,        // variabile per la curtosi
   int&          error_code       // variabile per memorizzare il codice errore
   );
```

Parametri

mu

[in] parametro della media della distribuzione (valore atteso).

sigma

[in] parametro sigma della distribuzione (deviazione radice-quadrata-media).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se i momenti sono stati calcolati con successo, altrimenti false.

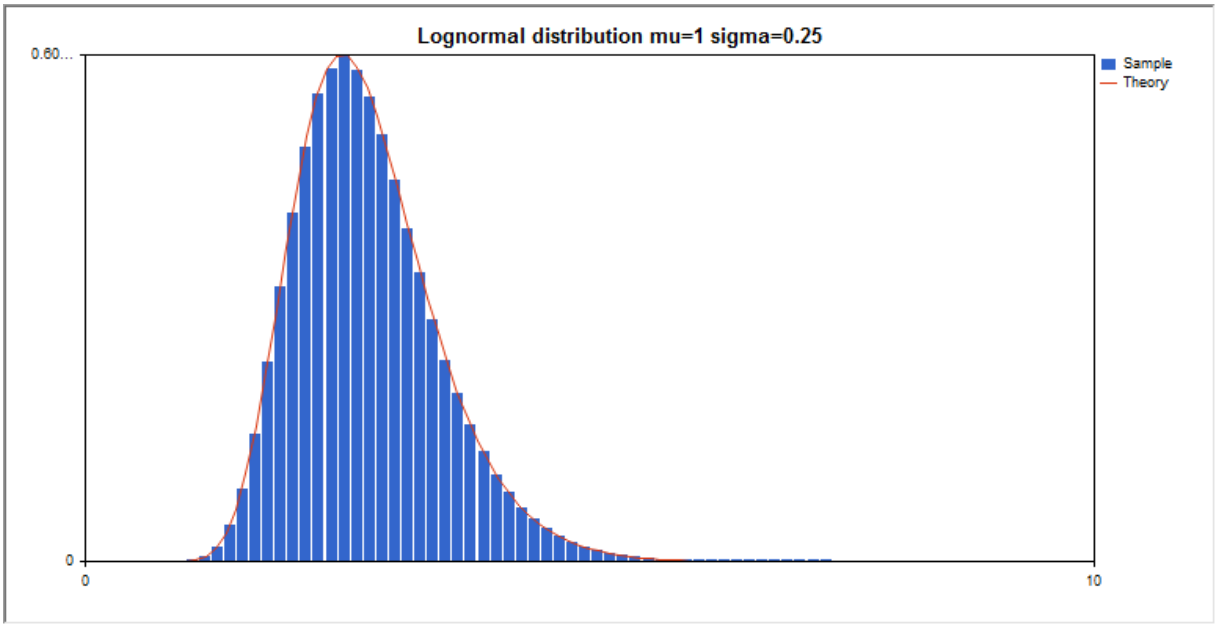
Distribuzione log-normale

Questa sezione contiene le funzioni per lavorare con la distribuzione log-normale. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge log-normale. La distribuzione log-normale è definita dalla seguente formula:

$$f_{Lognormal}(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

dove:

- x — valore della variabile casuale
- μ — logaritmo del valore atteso
- Σ — logaritmo della deviazione radice-quadratica-media



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityLognormal	Calcola la funzione di densità di probabilità della distribuzione log-normale
MathCumulativeDistributionLognormal	Calcola il valore della funzione di distribuzione di probabilità lognormale
MathQuantileLognormal	Calcola il valore della funzione di distribuzione lognormale inversa per la probabilità specificata
MathRandomLognormal	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge log-normale
MathMomentsLognormal	Calcola i valori numerici teoriche dei primi 4 momenti della distribuzione log-normale

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Lognormal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double mean_value=1.0;  // logaritmo di un valore atteso (log mean)
input double std_dev=0.25;    // logaritmo della deviazione della radice-quadatica-mec
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // il numero di valori nell'esempio
   int ncells=51;       // il numero di intervalli nell'istogramma
   double x[];          // centro degli intervalli dell'istogramma
   double y[];          // il numero di valori dall'esempio che cade all'interno dell
   double data[];       // esempio di valori casuali
   double max,min;      // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione log-normale
   MathRandomLognormal(mean_value,std_dev,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLognormal(x2,mean_value,std_dev,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Lognormal distribution mu=%G sigma=%G",mean_va
   graphic.BackgroundMainSize(16);
//--- disabilita la scalatura automatica dell'asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityLognormal

Calcola il valore della funzione di densità di probabilità di distribuzione log-normale con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // logaritmo del valore atteso (log mean)
   const double  sigma,          // logaritmo della deviazione radice-quadrata-media
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione log-normale con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // logaritmo del valore atteso (log mean)
   const double  sigma,          // logaritmo della deviazione radice-quadrata-media
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione log-normale con i parametri mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce NaN. Analogo di dlnorm() in R.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // array con i valori della variabile random
   const double   mu,            // logaritmo del valore atteso (log mean)
   const double   sigma,         // logaritmo della deviazione radice-quadrata-media
   const bool     log_mode,      // calcola il logaritmo del valore, se log_mode=true
   double&        result[]       // array per i valori della funzione di densità di p
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione log-normale con i parametri mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // array con i valori della variabile random
   const double   mu,            // logaritmo del valore atteso (log mean)
   const double   sigma,         // logaritmo della deviazione radice-quadrata-media
   double&        result[]       // array per i valori della funzione di densità di p
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] Logaritmo del valore atteso (log_mean).

sigma

[in] Logaritmo della deviazione radice-quadrata-media (log deviazione standard).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per ottenere i valori della funzione di densità di probabilità.

MathCumulativeDistributionLognormal

Calcola la funzione di distribuzione log-normale di probabilità con i parametri di mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // logaritmo del valore atteso (log mean)
   const double  sigma,          // logaritmo della deviazione radice-quadrata-media
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione log-normale di probabilità con i parametri di mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valore della variabile casuale
   const double  mu,             // logaritmo del valore atteso (log mean)
   const double  sigma,          // logaritmo della deviazione radice-quadrata-media
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione log-normale di probabilità con i parametri di mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di plnorm() in R.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // array con i valori della variabile random
   const double   mu,            // logaritmo del valore atteso (log mean)
   const double   sigma,         // logaritmo della deviazione radice-quadrata-media
   const bool     tail,          // flag di calcolo, se true, allora viene calcolata
   const bool     log_mode,      //flag di calcolo del logaritmo del valore, se log_
   double&        result[]       // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione log-normale di probabilità con i parametri di mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // array con i valori della variabile random
   const double   mu,            // logaritmo del valore atteso (log mean)
   const double   sigma,         // logaritmo della deviazione radice-quadrata-media
   double&        result[]       // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] Logaritmo del valore atteso (log_mean).

sigma

[in] Logaritmo della deviazione radice-quadrata-media (log deviazione standard).

tail

[in] Flag di calcolo, se è true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per ottenere i valori della funzione di probabilità.

MathQuantileLognormal

Per la probabilità specificata, la funzione calcola il valore della funzione di distribuzione inversa log-normale con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  mu,             // logaritmo del valore atteso (log mean)
   const double  sigma,          // logaritmo della deviazione radice-quadrata-media
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la probabilità specificata, la funzione calcola il valore della funzione di distribuzione inversa log-normale con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  mu,             // logaritmo del valore atteso (log mean)
   const double  sigma,          // logaritmo della deviazione radice-quadrata-media
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array probability[ ] di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione log-normale con i parametri mu e sigma. In caso di errore restituisce false. Analogo diqlnorm() in R.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // array con i valori di probabilità della variabile
   const double   mu,             // logaritmo del valore atteso (log mean)
   const double   sigma,          // logaritmo della deviazione radice-quadrata-media
   const bool     tail,           // flag di calcolo, se false, allora il calcolo vier
   const bool     log_mode,       // flag di calcolo, se log_mode=true, il calcolo vie
   double&        result[]        // array con i valori dei quantili
   );
```

Per lo specificato array probability[ ] di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione log-normale con i parametri mu e sigma. In caso di errore restituisce false.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // array con i valori di probabilità della variabile
   const double   mu,             // logaritmo del valore atteso (log mean)
   const double   sigma,          // logaritmo della deviazione radice-quadrata-media
   double&        result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità di accadimento variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

mu

[in] Logaritmo del valore atteso (log_mean).

sigma

[in] Logaritmo della deviazione radice-quadrata-media (log deviazione standard).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomLognormal

Genera una variabile pseudocasuale distribuita secondo la legge log-normale con i parametri sigma mu. In caso di errore restituisce NaN.

```
double  MathRandomLognormal(
   const double  mu,              // logaritmo del valore atteso (log mean)
   const double  sigma,           // logaritmo della deviazione radice-quadrata-media
   int&          error_code       // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge log-normale con i parametri sigma mu. In caso di errore restituisce false. Analogo di rlnorm() in R.

```
double  MathRandomLognormal(
   const double  mu,              // logaritmo del valore atteso (log mean)
   const double  sigma,           // logaritmo della deviazione radice-quadrata-media
   const int     data_count,      // ammontare di dati richiesti
   double&       result[]         // array con i valori di variabili pseudocasuali
   );
```

Parametri

mu

[in] Logaritmo del valore atteso (log_mean).

sigma

[in] Logaritmo della deviazione radice-quadrata-media (log deviazione standard).

data_count

[in] Ammontare di dati richiesti.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array con i valori di variabili pseudocasuali.

MathMomentsLognormal

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione log-normale. Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

```
double  MathMomentsLognormal(
   const double  mu,             // logaritmo del valore atteso (log mean)
   const double  sigma,          // logaritmo della deviazione radice-quadrata-media
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per memorizzare l' error code
   );
```

Parametri

mu

[in] Logaritmo del valore atteso (log_mean).

sigma

[in] Logaritmo della deviazione radice-quadrata-media (log deviazione standard).

mean

[in] Variabile per la media.

variance

[out] Variabile per la varianza.

skewness

[out] Variabile per l'asimmetria.

kurtosis

[out] Variabile per la curtosi.

error code

[out] variabile per memorizzare il codice di errore.

Valore di ritorno

Restituisce true se i momenti sono stati calcolati con successo, altrimenti false.

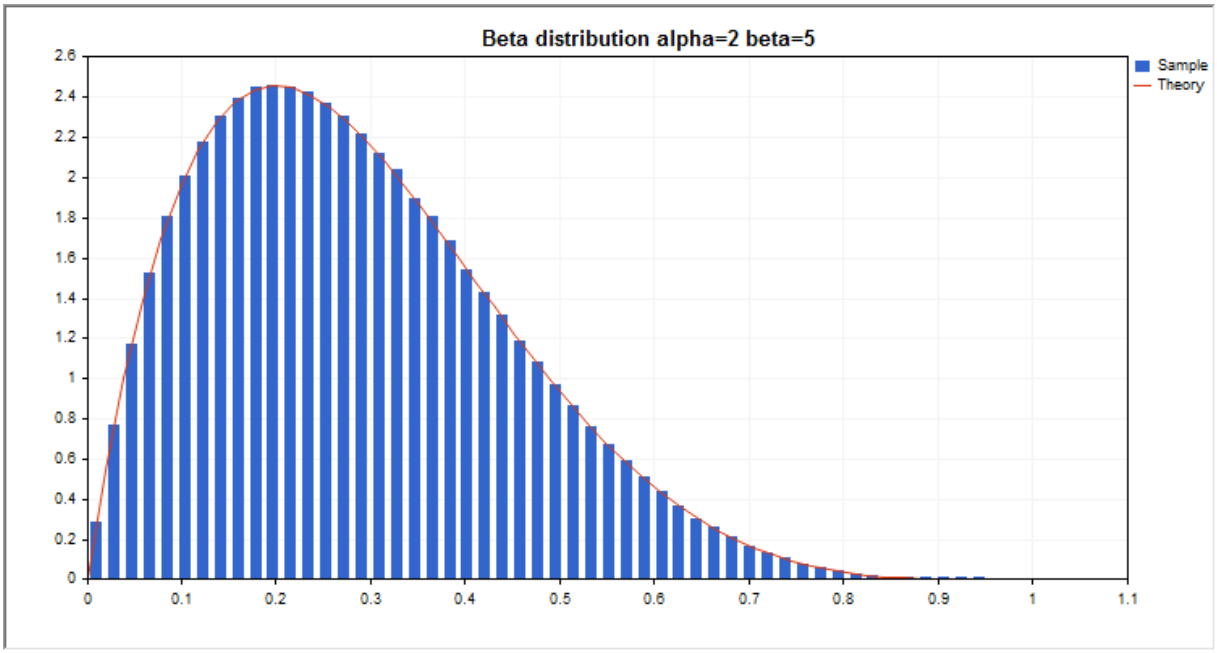
Distribuzione Beta

Questa sezione contiene le funzioni per lavorare con la distribuzione beta. Esse consentono di calcolare densità, probabilità, quantili e di generare numeri pseudo casuali distribuiti secondo la legge corrispondente. La distribuzione beta è definita dalla seguente formula:

$$f_{Beta}(x \mid a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

dove:

- x — valore della variabile casuale
- a — il primo parametro della distribuzione beta
- b — il secondo parametro della distribuzione beta



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityBeta	Calcola la funzione di densità di probabilità della distribuzione beta
MathCumulativeDistributionBeta	Calcola il valore della funzione di distribuzione di probabilità beta
MathQuantileBeta	Calcola il valore della funzione di distribuzione beta inversa per la probabilità specificata
MathRandomBeta	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione beta
MathMomentsBeta	Calcola i valori numerici teorich dei primi 4 momenti della distribuzione beta

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Beta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double alpha=2;   // il primo parametro della distribuzione beta (shape1)
input double beta=5;    // il secondo parametro della distribuzione beta (shape2)
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // il numero di valori nell'esempio
   int ncells=51;          // il numero di intervalli nell'istogramma
   double x[];             // centro degli intervalli dell'istogramma
   double y[];             // il numero di valori dall'esempio che cade all'interno dell
   double data[];          // esempio di valori casuali
   double max,min;         // i valori massimo e minimo nell'esempio
//--- ottenere un esempio(campione) dalla distribuzione beta
   MathRandomBeta(alpha,beta,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityBeta(x2,alpha,beta,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Beta distribution alpha=%G beta=%G",alpha,beta
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityBeta

Calcola il valore della funzione di densità di probabilità della distribuzione beta con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione beta con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione beta con i parametri A e B per una serie di variabili casuali x[]. In caso di errore restituisce false. Analogo di dbeta() in R.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (fo
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se lo
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione beta con i parametri A e B per una serie di variabili casuali x[]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (fo
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Il primo parametro della distribuzione beta (forma 1).

b

[in] Il secondo parametro della distribuzione beta (forma 2)

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionBeta

Calcola la funzione di distribuzione di probabilità di distribuzione beta con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (forr
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione beta con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (forr
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione beta con i parametri A e B per una serie di variabili casuali x [ ]. In caso di errore restituisce false. Analogo di pbeta() in R.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione beta con i parametri A e B per una serie di variabili casuali x [ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Il primo parametro della distribuzione beta (forma 1).

b

[in] Il secondo parametro della distribuzione beta (forma 2)

tail

[in] Flag di calcolo, se è true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileBeta

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione beta inversa con parametri a e b. In caso di errore restituisce NaN.

```
double  MathQuantileBeta(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione beta inversa con parametri a e b. In caso di errore restituisce NaN.

```
double  MathQuantileBeta(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità[]* array di valori di probabilità, la funzione calcola i valori della funzione di distribuzione beta inversa con i parametri a e b. In caso di errore restituisce false. Analogo di qbeta() in R.

```
double  MathQuantileBeta(
   const double&  probability[], // array co i valori della prababilità di variabile r
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per la specificata *probabilità[]* array di valori di probabilità, la funzione calcola i valori della funzione di distribuzione beta inversa con i parametri a e b. In caso di errore restituisce false.

```
bool  MathQuantileBeta(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

a

[in] Il primo parametro di distribuzione beta (shape1).

b

[in] Il secondo parametro di distribuzione beta (shape2).

tail

[in] Flag di calcolo, se lower_tail=false, quindi il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomBeta

Genera una variabile pseudocasuale distribuita secondo la legge della distribuzione beta con parametri a e b. In caso di errore restituisce NaN.

```
double  MathRandomBeta(
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuiti secondo la legge della distribuzione beta con parametri a e b. In caso di errore restituisce false. Analogo di rbeta() in R.

```
bool  MathRandomBeta(
   const double  a,               // il primo parametro della distribuzione beta (forma
   const double  b,               // il secondo parametro della distribuzione beta (fo
   const int     data_count,      // ammontare dei dati richiesti
   double&       result[]         // array per ottenere le variabili pseudocasuali
   );
```

Parametri

a

[in] Il primo parametro di distribuzione beta (shape1)

b

[in] Il secondo parametro di distribuzione beta (shape2).

data_count

[in] Il numero di variabili pseudocasuali da ottenere.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsBeta

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione beta

```
double  MathMomentsBeta(
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

a

[in] Il primo parametro della distribuzione beta (shape1).

b

[in] Il secondo parametro della distribuzione beta (shape2).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se i momenti sono stati calcolati con successo, altrimenti false.

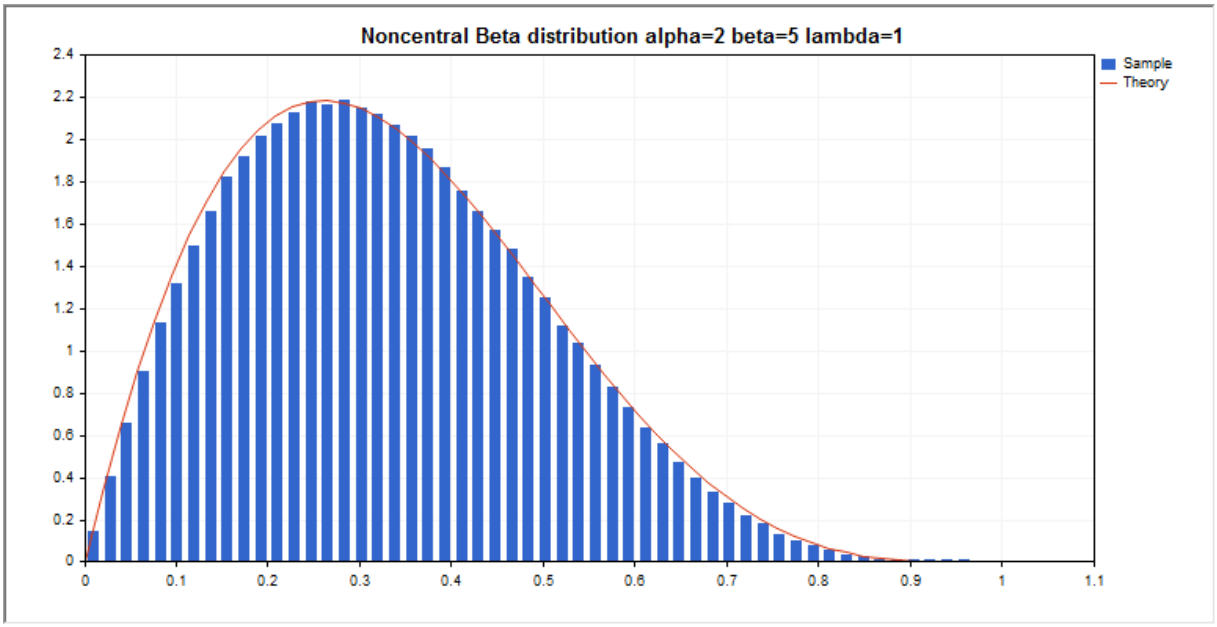
Distribuzione beta non centrale

Questa sezione contiene le funzioni per lavorare con distribuzione beta non centrale. Esse consentono di calcolare densità, probabilità, quantili e di generare numeri pseudo casuali distribuiti secondo la legge corrispondente. La distribuzione beta non centrale è definita dalla seguente formula:

$$f_{NoncentralBeta}(x|a,b,\lambda)=\sum_{r=0}^{\infty}e^{-\frac{\lambda}{2}}\frac{\left(\frac{\lambda}{2}\right)^r}{r!}\frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r,b)}$$

dove:

- x — valore della variabile casuale
- a — il primo parametro della distribuzione beta
- b — il secondo parametro della distribuzione beta
- λ — parametro di non centralità



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityNoncentralBeta	Calcola la funzione di densità di probabilità della distribuzione beta non centrale
MathCumulativeDistributionNoncentralBeta	Calcola il valore della funzione di distribuzione di probabilità non centrale beta
MathQuantileNoncentralBeta	Calcola il valore della funzione di distribuzione beta non centrale inversa per la probabilità specificata
MathRandomNoncentralBeta	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione beta non centrale

Funzione	Descrizione
MathMomentsNoncentralBeta	Calcola i valori numerici teoriche dei primi 4 momenti della distribuzione beta non centrale

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralBeta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double a_par=2;     // il primo parametro della distribuzione beta (shape1)
input double b_par=5;     // il secondo parametro della distribuzione beta (shape2)
input double l_par=1;     // parametro di non centralità (lambda)
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // il numero di valori nell'esempio
   int ncells=53;        // il numero di intervalli nell'istogramma
   double x[];           // centro degli intervalli dell'istogramma
   double y[];           // il numero di valori dall'esempio che cade all'interno dell
   double data[];        // esempio di valori casuali
   double max,min;       // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione beta non centrale
   MathRandomNoncentralBeta(a_par,b_par,l_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralBeta(x2,a_par,b_par,l_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral Beta distribution alpha=%G beta=%G
                          a_par,b_par,l_par));
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
```

```
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralBeta

Calcola il valore della funzione di densità di probabilità di distribuzione beta non centrale con i parametri A, B e lambda per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (forr
   const double  lambda,         // parametro di noncentralità
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione beta non centrale con i parametri A, B e lambda per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (forr
   const double  lambda,         // parametro di noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione beta non centrale con i parametri A, B e lambda per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dbeta() in R.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (fo
   const double  lambda,         // parametro di noncentralità
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se lo
   double&       result[]        // array per i valori della funzione di densita di p
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione beta con i parametri A e B per una serie di variabili casuali x[]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (fo
   const double  lambda,         // parametro di noncentralità
   double&       result[]        // array per i valori della funzione di densita di p
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Il primo parametro della distribuzione beta (forma 1).

b

[in] Il secondo parametro della distribuzione beta (forma 2)

lambda

[in] Parametro noncentralità

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionNoncentralBeta

Calcola la funzione di distribuzione di probabilità di distribuzione beta non centrale con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (forr
   const double  lambda,         // parametro di noncentralità
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione beta non centrale con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (forr
   const double  lambda,         // parametro di noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione beta non centrale con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pbeta() in R.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const double  lambda,         // parametro di noncentralità
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione beta non centrale con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const double  lambda,         // parametro di noncentralità
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Il primo parametro della distribuzione beta (forma 1).

b

[in] Il secondo parametro della distribuzione beta (forma 2)

lambda

[in] Parametro noncentralità

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileNoncentralBeta

Calcola il valore della funzione di distribuzione di probabilità inversa della distribuzione beta non centrale con i parametri A, B e lambda per la probabilità di occorrenza di una variabile casuale *probabilità*. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const double  lambda,         // parametro di noncentralità
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità inversa della distribuzione beta non centrale con i parametri A, B e lambda per la probabilità di occorrenza di una variabile casuale *probabilità*. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const double  lambda,         // parametro di noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array probability[ ] di valori di probabilità, la funzione calcola il valore della funzione di distribuzione di probabilità inversa della distribuzione beta non centrale con i parametri A, B e lambda. In caso di errore restituisce false. Analogo di qbeta() in R.

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // array con i valori di probabilità della variabile
   const double  a,               // il primo parametro della distribuzione beta (forma
   const double  b,               // il secondo parametro della distribuzione beta (for
   const double  lambda,          // parametro di noncentralità
   const bool    tail,            // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,        // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]         // array con i valori dei quantili
   );
```

Per lo specificato array probability[ ] di valori di probabilità, la funzione calcola il valore della funzione di distribuzione di probabilità inversa della distribuzione beta non centrale con i parametri A, B e lambda. In caso di errore restituisce false.

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
```

```
   const double  lambda,          // parametro di noncentralità
   double&       result[]         // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

a

[in] Il primo parametro di distribuzione beta (shape1).

b

[in] Il secondo parametro di distribuzione beta (shape2).

lambda

[in] Parametro noncentralità.

tail

[in] Flag di calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomNoncentralBeta

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione beta non centrale A, B e parametri lambda. In caso di errore restituisce NaN.

```
double  MathRandomNoncentralBeta(
   const double  a,             // il primo parametro della distribuzione beta (forma1
   const double  b,             // il secondo parametro della distribuzione beta (forr
   const double  lambda,        // parametro di noncentralità
   int&          error_code     // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione beta non centrale con i parametri A, B e lambda. In caso di errore restituisce false. Analogo di rbeta() in R.

```
bool  MathRandomNoncentralBeta(
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const double  lambda,         // parametro di noncentralità
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array per ottenere le variabili pseudocasuali
   );
```

Parametri

a

[in] Il primo parametro di distribuzione beta (shape1)

b

[in] Il secondo parametro di distribuzione beta (shape2).

lambda

[in] Parametro noncentralità

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsNoncentralBeta

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione beta non centrale con i parametri A, B e lambda.

```
double  MathMomentsNoncentralBeta(
   const double  a,              // il primo parametro della distribuzione beta (forma
   const double  b,              // il secondo parametro della distribuzione beta (for
   const double  lambda,         // parametro di noncentralità
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

a

[in] Il primo parametro della distribuzione beta (shape1).

b

[in] Il secondo parametro della distribuzione beta (shape2).

lambda

[in] Parametro noncentralità

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

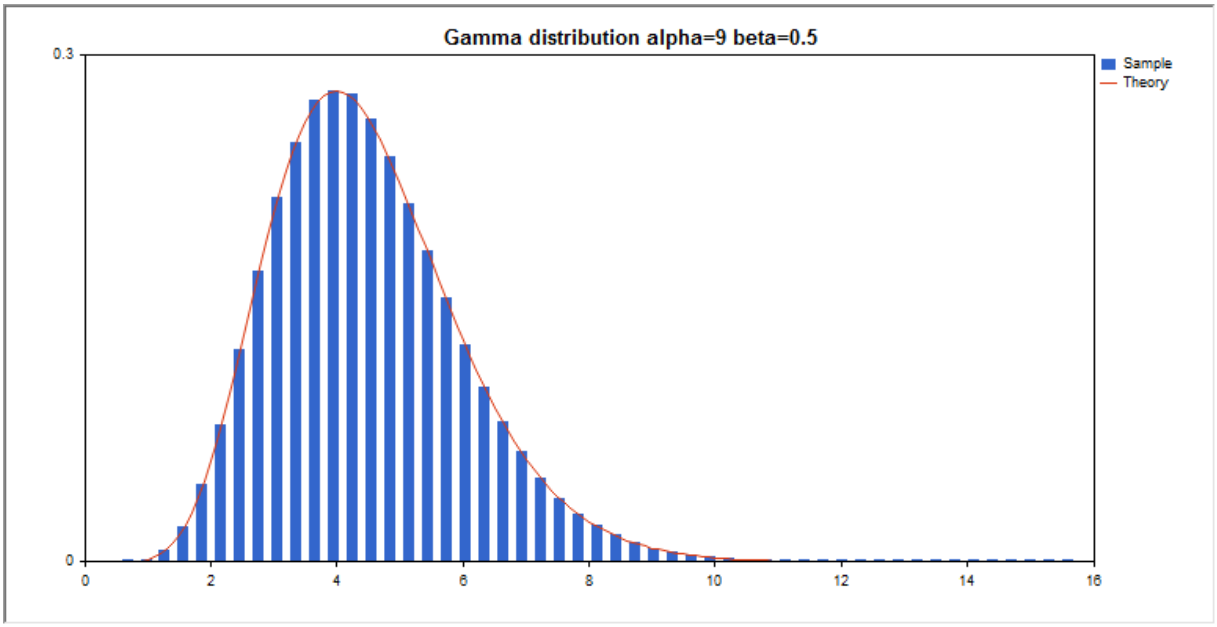
Distribuzione gamma

Questa sezione contiene funzioni per lavorare con distribuzione gamma. Esse consentono di calcolare densità, probabilità, quantili e di generare numeri pseudo casuali distribuiti secondo la legge corrispondente. La distribuzione gamma è definita dalla seguente formula:

$$f_{Gamma}(x|a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

dove:

- x — valore della variabile casuale
- a — il primo parametro della distribuzione
- b — il secondo parametro della distribuzione



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityGamma	Calcola la funzione di densità di probabilità della distribuzione gamma
MathCumulativeDistributionGamma	Calcola il valore della funzione di distribuzione di probabilità gamma
MathQuantileGamma	Calcola il valore della funzione di distribuzione gamma inversa per la probabilità specificata
MathRandomGamma	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione gamma
MathMomentsGamma	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione gamma

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Gamma.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double alpha=9;   // il primo parametro della distribuzione (shape)
input double beta=0.5;  // il secondo parametro di distribuzione (scale)
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // il numero di valori nell'esempio
   int ncells=51;        // il numero di intervalli nell'istogramma
   double x[];           // centro degli intervalli dell'istogramma
   double y[];           // il numero di valori dall'esempio che cade all'interno dell
   double data[];        // esempio di valori casuali
   double max,min;       // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione gamma
   MathRandomGamma(alpha,beta,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityGamma(x2,alpha,beta,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Gamma distribution alpha=%G beta=%G",alpha,bet
   graphic.BackgroundMainSize(16);
//--- disabilita la scalatura automatica dell'asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,1));
   graphic.YAxis().Min(0);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityGamma

Calcola il valore della funzione di densità di probabilità della distribuzione gamma con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione gamma con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione gamma con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dgamma() in R.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione gamma con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Il primo parametro della distribuzione (forma).

b

[in] Il secondo parametro della distribuzione (scala).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

>MathCumulativeDistributionGamma

Calcola la funzione di distribuzione di probabilità di distribuzione gamma con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione gamma con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // valore di variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione gamma con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pgamma() in R.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione gamma con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Il primo parametro della distribuzione (forma).

b

[in] Il secondo parametro della distribuzione (scala)

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileGamma

Per la specificata *probabilità*, la funzione calcola il valore della funzione inversa distribuzione gamma con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileGamma(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione inversa distribuzione gamma con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileGamma(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probabilità[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione gamma con i parametri A e B. In caso di errore restituisce false. Analogo di qgamma() in R.

```
double  MathQuantileGamma(
   const double& probability[],  // array i valori della probabilità della variabile
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probabilità[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione gamma con i parametri A e B. In caso di errore restituisce false.

```
bool  MathQuantileGamma(
   const double& probability[],  // array i valori della probabilità della variabile
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

a

[in] Il primo parametro della distribuzione (forma).

b

[in] Il secondo parametro della distribuzione (scala).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomGamma

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione gamma con i parametri a e b. In caso di errore restituisce NaN.

```
double  MathRandomGamma(
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione gamma con i parametri a e b. In caso di errore restituisce false. Analogo dirgamma() in R.

```
bool  MathRandomGamma(
   const double  a,               // il primo parametro della distribuzione (forma)
   const double  b,               // il secondo parametro della distribuzione (scala)
   const int     data_count,      // ammontare dei dati richiesti
   double&       result[]         // array con i valori delle variabili pseudocasuali
   );
```

Parametri

a

[in] Il primo parametro della distribuzione (forma).

b

[in] Il secondo parametro della distribuzione (scala).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsGamma

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione gamma con i parametri A e B.

```
double  MathMomentsGamma(
   const double  a,              // il primo parametro della distribuzione (forma)
   const double  b,              // il secondo parametro della distribuzione (scala)
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

a

[in] Il primo parametro della distribuzione (forma).

b

[in] Il secondo parametro della distribuzione (scala).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

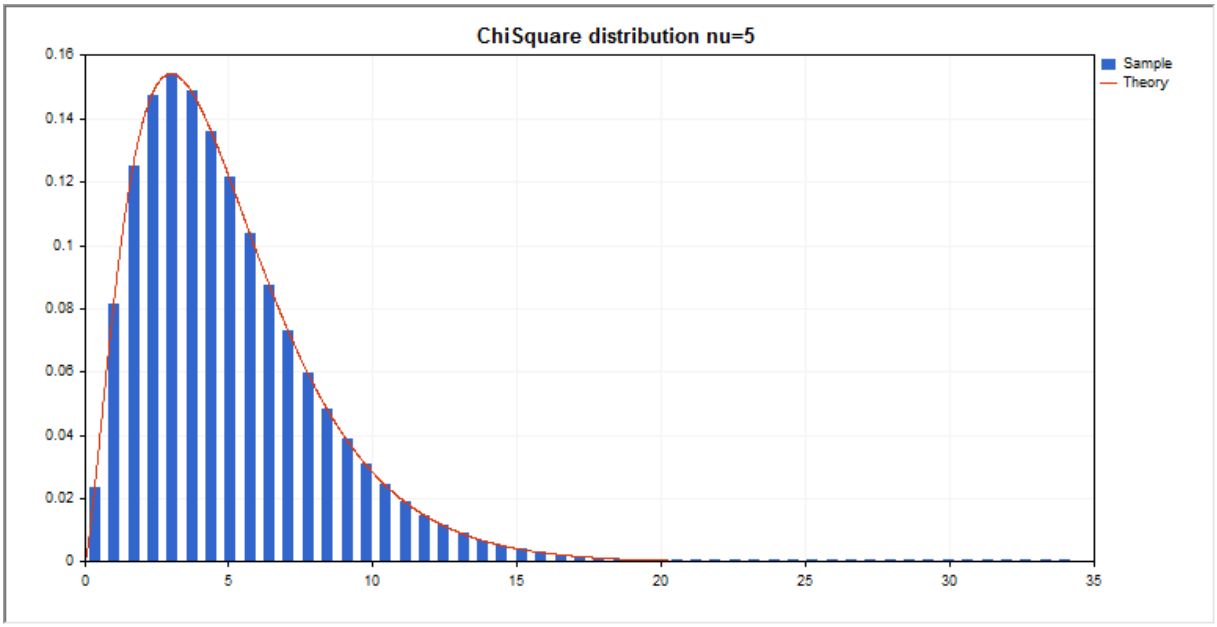
Distribuzione del chi quadrato

Questa sezione contiene funzioni per lavorare con distribuzione del chi quadrato. Esse consentono di calcolare densità, probabilità, quantili e di generare numeri pseudo casuali distribuiti secondo la legge corrispondente. La distribuzione chi-quadrato è definita dalla seguente formula:

$$f_{Chi-Square}(x|\nu) = \frac{x^{\frac{(\nu-2)}{2}} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}$$

dove:

- x — valore della variabile casuale
- ν — il numero di gradi di libertà



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityChiSquare	Calcola la funzione di densità di probabilità della distribuzione del chi quadrato
MathCumulativeDistributionChiSquare	Calcola il valore della funzione di distribuzione di probabilità chi-quadrato
MathQuantileChiSquare	Calcola il valore della funzione di distribuzione chi-quadrato inversa per la probabilità specificata
MathRandomChiSquare	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione del chi-quadrato

Funzione	Descrizione
MathMomentsChiSquare	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione chi-quadrato

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\ChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double nu_par=5;     // il numero di gradi di libertà
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // il numero di valori nell'esempio
   int ncells=51;        // il numero di intervalli nell'istogramma
   double x[];           // centro degli intervalli dell'istogramma
   double y[];           // il numero di valori dall'esempio che cade all'interno dell
   double data[];        // esempio di valori casuali
   double max,min;       // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione del chi-quadrato
   MathRandomChiSquare(nu_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityChiSquare(x2,nu_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
```

```
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("ChiSquare distribution nu=%G ",nu_par));
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityChiSquare

Calcola il valore della funzione di densità di probabilità di distribuzione del chi-quadrato con il parametro NU per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione del chi-quadrato con il parametro NU per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione chi-quadrato con il parametro NU per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dchisq() in R.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione chi-quadrato con il parametro NU per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro di distribuzione (numero di gradi di libertà)

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionChiSquare

Calcola la funzione di distribuzione di probabilità di distribuzione del chi-quadrato con il parametro NU per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione del chi-quadrato con il parametro NU per una variabile casuale X. In caso di errore restituisce NaN

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione del chi-quadrato con il parametro NU per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pchisq() in R.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione del chi-quadrato con il parametro NU per una serie di variabili casuali x [ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileChiSquare

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione del chi-quadrato inverso. In caso di errore restituisce NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione del chi-quadrato inverso. In caso di errore restituisce NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione del chi-quadrato inverso. In caso di errore restituisce false. Analogo di qchisq() in R.

```
double  MathQuantileChiSquare(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione del chi-quadrato inverso. In caso di errore restituisce false.

```
bool  MathQuantileChiSquare(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  nu,             // parametro della distribuzione (numero di gradi di
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomChiSquare

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione del chi-quadrato con il parametro NU. In caso di errore restituisce NaN.

```
double  MathRandomChiSquare(
   const double  nu,            // parametro della distribuzione (numero di gradi di l
   int&          error_code     // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione del chi-quadrato con il parametro NU. In caso di errore restituisce false. Analogo di rchisq() in R.

```
bool  MathRandomChiSquare(
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsChiSquare

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione chi-quadrato con il parametro NU.

```
double  MathMomentsChiSquare(
   const double  nu,              // parametro della distribuzione (numero di gradi di
   double&       mean,            // variabile per la media
   double&       variance,        // variabile per la varianza
   double&       skewness,        // variabile per l'asimmetria
   double&       kurtosis,        // variabile per la curtosi
   int&          error_code       // variabile per il codice errore
   );
```

Parametri

nu

[in] Parametro di distribuzione (numero di gradi di libertà).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

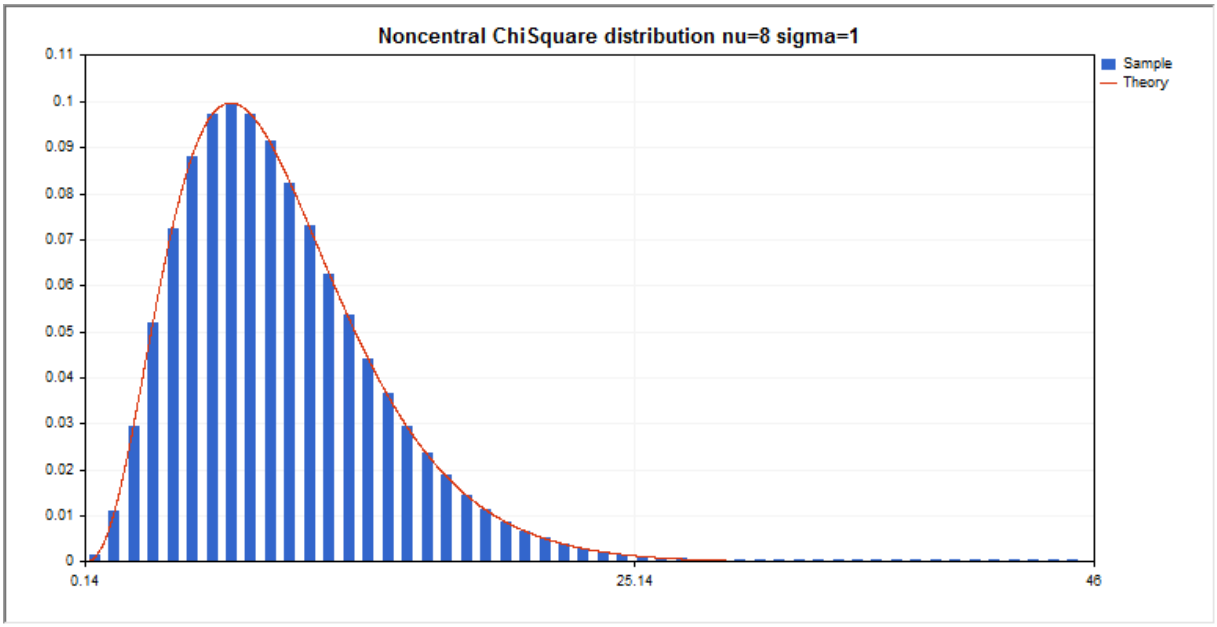
Distribuzione chi-quadrato non centrale

Questa sezione contiene funzioni per lavorare con la distribuzione del chi-quadrato non centrale. Esse consentono di calcolare densità, probabilità, quantili e di generare numeri pseudo casuali distribuiti secondo la legge corrispondente. La distribuzione chi-quadrato non centrale è definita dalla seguente formula:

$$f_{NoncentralChiSquare}(x|\nu,\sigma) = \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)} x^{\frac{\nu}{2}-1} e^{-\frac{(x+\sigma)}{2}} \sum_{r=0}^{\infty} \frac{(\lambda x)^r}{(2r)!} \frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

dove:

- x — valore della variabile casuale
- ν — il numero di gradi di libertà
- σ — parametro non centralità



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityNoncentralChiSquare	Calcola la funzione di densità di probabilità della distribuzione chi-quadrato non centrale
MathCumulativeDistributionNoncentralChiSquare	Calcola il valore della funzione di distribuzione di probabilità chi-quadrato non centrale
MathQuantileNoncentralChiSquare	Calcola il valore della funzione di distribuzione chi-quadrato non centrale inversa per la probabilità specificata

Funzione	Descrizione
MathRandomNoncentralChiSquare	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione del chi-quadrato
MathMomentsNoncentralChiSquare	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione chi-quadrato non centrale

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double nu_par=8;     // il numero di gradi di libertà
input double si_par=1;     // parametro non centralità
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // il numero di valori nell'esempio
   int ncells=51;          // il numero di intervalli nell'istogramma
   double x[];             // centro degli intervalli dell'istogramma
   double y[];             // il numero di valori dall'esempio che cade all'interno dell
   double data[];          // esempio di valori casuali
   double max,min;         // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione chi-quadrato non centrale
   MathRandomNoncentralChiSquare(nu_par,si_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralChiSquare(x2,nu_par,si_par,false,y2);
//--- imposta la scala
```

```
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral ChiSquare distribution nu=%G sigma=
   graphic.BackgroundMainSize(16);
//--- disabilitare scalatura automatica dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(NormalizeDouble(max,0));
   graphic.XAxis().Min(min);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
```

```
      {
       int ind=int((data[i]-minv)/width);
       if(ind>=cells) ind=cells-1;
       frequency[ind]++;
      }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralChiSquare

Calcola il valore della funzione di densità di probabilità di distribuzione chi-quadrato non centrale con i parametri nu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  sigma,          // parametro noncentralità
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione chi-quadrato non centrale con i parametri nu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  sigma,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione chi quadrato non centrale con i parametri nu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dchisq() in R.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione chi-quadrato con il parametro NU per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionNoncentralChiSquare

Calcola la funzione di distribuzione di probabilità di distribuzione chi quadrato non centrale con i parametri nu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  sigma,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se lower_tail=true, allora viene c
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione chi quadrato non centrale con i parametri nu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  sigma,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione chi quadrato non centrale con i parametri nu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pchisq() in R.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se lower_tail=true, allora viene
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione chi-quadrato non centrale con i parametri nu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileNoncentralChiSquare

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione inversa del chi-quadrato non centrale con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione inversa del chi-quadrato non centrale con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione del chi quadrato non centrale con i parametri nu e sigma. In caso di errore restituisce false. Analogo di qchisq() in R.

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // array con i valori di probabilità della variabile
   const double  nu,              // parametro della distribuzione (numero di gradi di
   const double  sigma,           // parametro noncentralità
   const bool    tail,            // flag di calcolo, se false, allora il calcolo vien
   const bool    log_mode,        // flag di calcolo, se log_mode=true, viene eseguito
   double&       result[]         // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione del chi-quadrato non centrale. In caso di errore restituisce false.

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomNoncentralChiSquare

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione chi-quadrato non centrale con i parametri nu e sigma. In caso di errore restituisce NaN.

```
double  MathRandomNoncentralChiSquare(
   const double  nu,            // parametro della distribuzione (numero di gradi di l
   const double  sigma,         // parametro noncentralità
   int&          error_code     // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione chi-quadrato non centrale con i parametri nu e sigma. In caso di errore restituisce false. Analogo di rchisq() in R.

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsNoncentralChiSquare

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione del chi-quadrato non centrale con i parametri nu e sigma.

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  sigma,          // parametro noncentralità
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

nu

[in] Parametro di distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

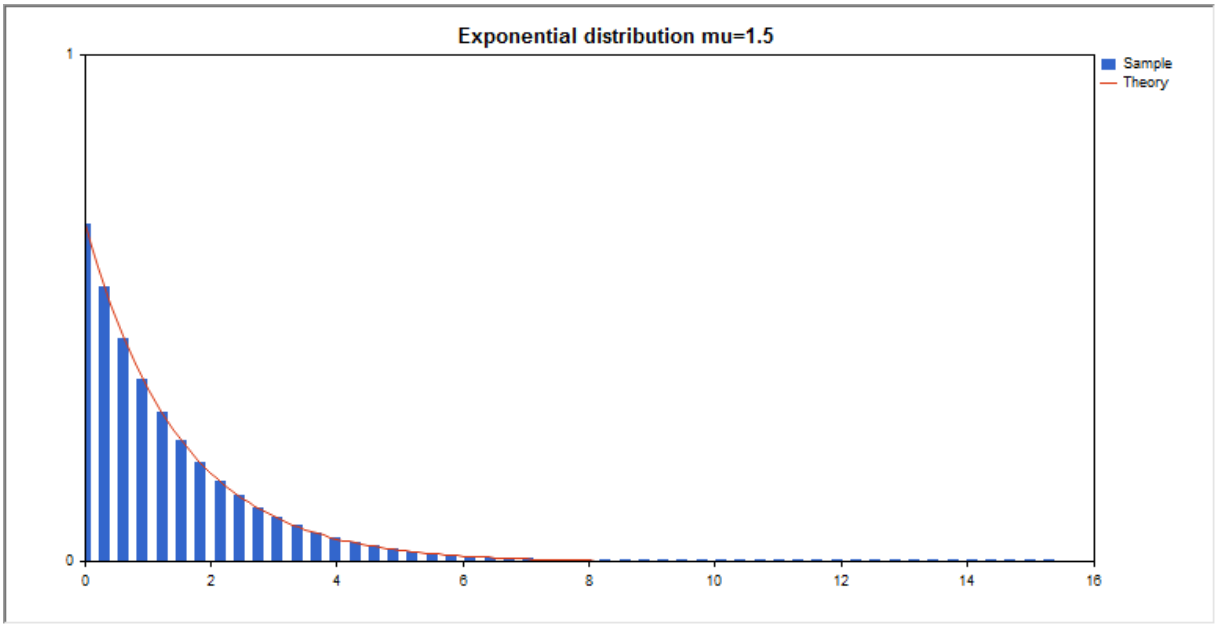
Esponenziale

Questa sezione contiene funzioni per lavorare con distribuzione esponenziale. Esse permettono di calcolare la densità, probabilità, quantili e di generare numeri pseudo-casuali distribuiti secondo la legge di distribuzione esponenziale. La distribuzione esponenziale è definita dalla seguente formula:

$$f_{Exponential}(x|\mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

dove:

- x — valore della variabile casuale
- μ — valore atteso



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityExponential	Calcola la funzione di densità di probabilità della distribuzione esponenziale
MathCumulativeDistributionExponential	Calcola il valore della funzione di distribuzione di probabilità esponenziale
MathQuantileExponential	Calcola il valore della funzione di distribuzione esponenziale inversa per la probabilità specificata
MathRandomExponential	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione esponenziale
MathMomentsExponential	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione esponenziale

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Exponential.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double mu_par=1.5;     // il numero di gradi di libertà
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // il numero di valori nell'esempio
   int ncells=51;          // il numero di intervalli nell'istogramma
   double x[];             // centro degli intervalli dell'istogramma
   double y[];             // il numero di valori dall'esempio che cade all'interno dell
   double data[];          // esempio di valori casuali
   double max,min;         // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione esponenziale
   MathRandomExponential(mu_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityExponential(x2,mu_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

```
   graphic.BackgroundMain(StringFormat("Exponential distribution mu=%G ",mu_par));
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
```

```
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityExponential

Calcola il valore della funzione di densità di probabilità della distribuzione esponenziale con parametro MU per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione esponenziale con parametro MU per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione esponenziale con il parametro MU per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo didexp() in R.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione esponenziale con il parametro MU per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] Parametro della distribuzione (valore atteso)

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionExponential

Calcola la funzione di distribuzione esponenziale della probabilità con parametro MU per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione esponenziale della probabilità con parametro MU per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità esponenziale con parametro MU per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pexp() in R.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità esponenziale con parametro MU per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro della distribuzione (valore atteso)
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] Parametro della distribuzione (valore atteso).

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileExponential

Per la specificata*probabilità*, la funzione calcola il valore della funzione inversa distribuzione esponenziale con parametro MU. In caso di errore restituisce NaN.

```
double  MathQuantileExponential(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  mu,             // parametro della distribuzione (valore atteso)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata*probabilità*, la funzione calcola il valore della funzione inversa distribuzione esponenziale con parametro MU. In caso di errore restituisce NaN.

```
double  MathQuantileExponential(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  mu,             // parametro della distribuzione (valore atteso)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione esponenziale con parametro MU. In caso di errore restituisce false. Analogo di qexp() in R.

```
double  MathQuantileExponential(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  mu,             // parametro della distribuzione (valore atteso)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione esponenziale con parametro MU. In caso di errore restituisce false.

```
bool  MathQuantileExponential(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  mu,             // parametro della distribuzione (valore atteso)
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

mu

[in] Parametro della distribuzione (valore atteso).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomExponential

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione esponenziale con il parametro MU. In caso di errore restituisce NaN.

```
double  MathRandomExponential(
   const double  mu,            // parametro della distribuzione (valore atteso)
   int&          error_code     // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione esponenziale con il parametro MU. In caso di errore restituisce false. Analogo di rexp() in R.

```
bool  MathRandomExponential(
   const double  mu,             // parametro della distribuzione (valore atteso)
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

mu

[in] Parametro della distribuzione (valore atteso).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsExponential

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione esponenziale con il parametro MU.

```
double  MathMomentsExponential(
   const double  mu,              // parametro della distribuzione (valore atteso)
   double&       mean,            // variabile per la media
   double&       variance,        // variabile per la varianza
   double&       skewness,        // variabile per l'asimmetria
   double&       kurtosis,        // variabile per la curtosi
   int&          error_code       // variabile per il codice errore
   );
```

Parametri

mu

[in] Parametro della distribuzione (valore atteso).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

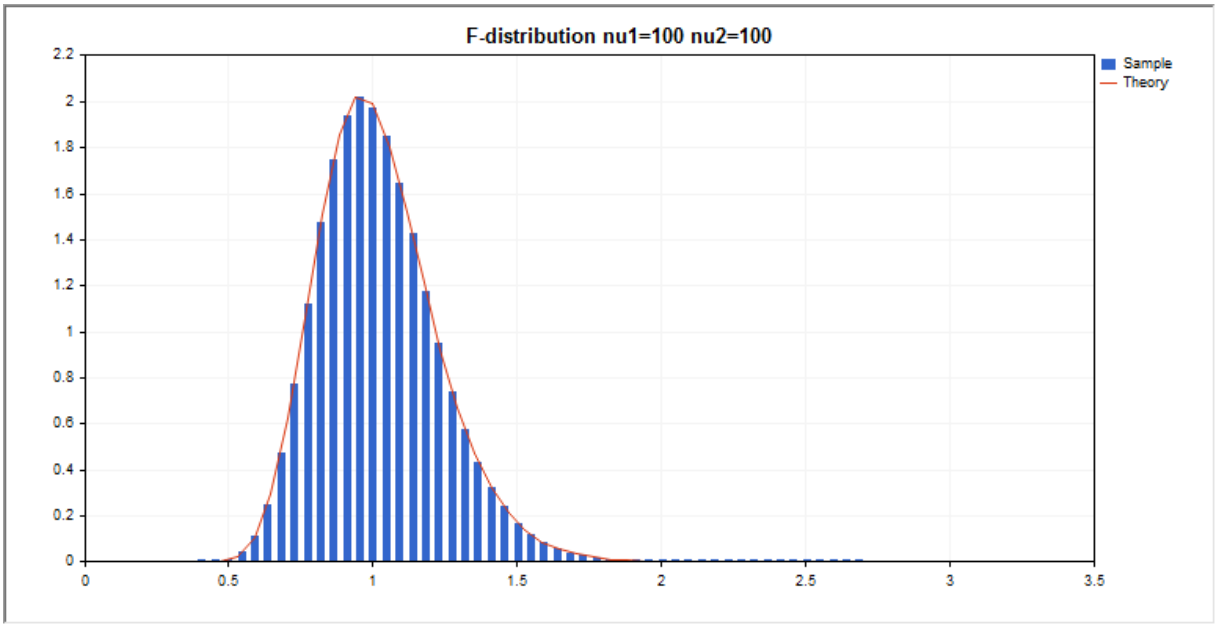
Distribuzione-F

Questa sezione contiene funzioni per lavorare con la distribuzione-F. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge di distribuzione-F di Fisher. La distribuzione-F è definita dalla seguente formula:

$$f_F(x|\nu_1,\nu_2)=\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}}\frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

dove:

- x — valore della variabile casuale
- ν1 — il primo parametro di distribuzione (numero di gradi di libertà)
- ν2 — il secondo parametro di distribuzione (numero di gradi di libertà)



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityF	Calcola la funzione di densità di probabilità della distribuzione-F
MathCumulativeDistributionF	Calcola il valore della funzione di distribuzione-F
MathQuantileF	Calcola il valore della funzione inversa distribuzione-F per la probabilità specificata
MathRandomF	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di Fisher

Funzione	Descrizione
MathMomentsF	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione-F di Fisher

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\F.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double nu_1=100;     // il primo numero di gradi di libertà
input double nu_2=100;     // il secondo numero di gradi di libertà
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // il numero di valori nell'esempio
   int ncells=51;        // il numero di intervalli nell'istogramma
   double x[];           // centro degli intervalli dell'istogramma
   double y[];           // il numero di valori dall'esempio che cade all'interno dell
   double data[];        // esempio di valori casuali
   double max,min;       // i valori massimo e minimo nell'esempio
// --- ottiene un campione da distribuzione-F di Fisher
   MathRandomF(nu_1,nu_2,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityF(x2,nu_1,nu_2,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("F-distribution nu1=%G nu2=%G",nu_1,nu_2));
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(4);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

```
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityF

Calcola il valore della funzione di densità di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di df() in R.

```
bool  MathProbabilityDensityF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionF

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pf() in R.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F di Fisher con i parametri NU1 e NU2 per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileF

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione-F inversa di Fisher con i parametri NU1 e NU2. In caso di errore restituisce NaN.

```
double  MathQuantileF(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione-F inversa di Fisher con i parametri NU1 e NU2. In caso di errore restituisce NaN.

```
double  MathQuantileF(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probabilità[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione-F inversa di Fisher con i parametri NU1 e NU2. In caso di errore restituisce false. Analogo di qf() in R.

```
double  MathQuantileF(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probabilità[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione-F inversa di Fisher con i parametri NU1 e NU2. In caso di errore restituisce false.

```
bool  MathQuantileF(
   const double& probability[],  // array i valori della probabilità della variabile
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

tail

[in] Flag di calcolo, se lower_tail=false, quindi il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomF

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione-F di Fisher con i parametri NU1 e NU2. In caso di errore restituisce NaN.

```
double  MathRandomF(
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   int&          error_code     // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione-F di Fisher con i parametri NU1 e NU2. In caso di errore restituisce false. Analogo dirf() in R.

```
bool  MathRandomF(
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsF

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione-F di Fisher con i parametri NU1 e NU2.

```
double  MathMomentsF(
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

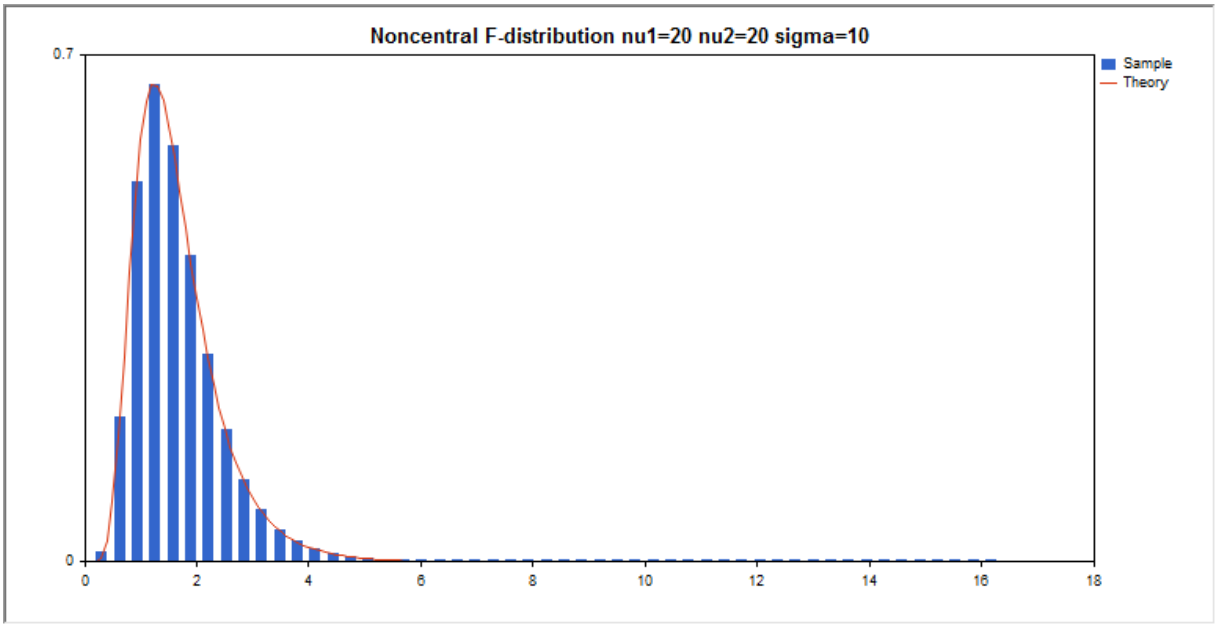
Distribuzione-F non centrale

Questa sezione contiene funzioni per lavorare con la distribuzione-F non centrale. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge di distribuzione-F non centrale di Fisher. La distribuzione-F non centrale è definita dalla seguente formula:

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma) = e^{-\frac{\sigma}{2}} \sum_{r=0}^{\infty} \frac{1}{r!}\left(\frac{\sigma}{2}\right)^r \frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)} \left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r} \frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

dove:

- x — valore della variabile casuale
- ν1 — il primo parametro di distribuzione (numero di gradi di libertà)
- ν2 — il secondo parametro di distribuzione (numero di gradi di libertà)
- σ — parametro non centralità



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityNoncentralF	Calcola la funzione di densità di probabilità della distribuzione-F non centrale
MathCumulativeDistributionNoncentralF	Calcola il valore della funzione di distribuzione-F non centrale
MathQuantileNoncentralF	Calcola il valore della funzione di distribuzione-F non centrale inversa per la probabilità specificata

Funzione	Descrizione
MathRandomNoncentralF	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione-F non centrale
MathMomentsNoncentralF	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione-F non centrale di Fisher

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralF.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double nu_1=20;     // il primo numero di gradi di libertà
input double nu_2=20;     // il secondo numero di gradi di libertà
input double sig=10;      // parametro di non centralità
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // il numero di valori nell'esempio
   int ncells=51;        // il numero di intervalli nell'istogramma
   double x[];           // centro degli intervalli dell'istogramma
   double y[];           // il numero di valori dall'esempio che cade all'interno dell
   double data[];        // esempio di valori casuali
   double max,min;       // i valori massimo e minimo nell'esempio
//--- ottiene un campione da distribuzione-F non centrale di Fisher
   MathRandomNoncentralF(nu_1,nu_2,sig,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralF(x2,nu_1,nu_2,sig,false,y2);
```

```
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral F-distribution nu1=%G nu2=%G sigma=
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralF

Calcola il valore della funzione di densità di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di g
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   const double  sigma,          // parametro noncentralità
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di g
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   const double  sigma,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di df() in R.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionNoncentralF

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   const double  sigma,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // valore di variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero di
   const double  sigma,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pf() in R.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero
   const double  sigma,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione di probabilità di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // array con i valori della variabile random
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero
   const double  sigma,          // parametro noncentralità
   double&       result[]        // array per i valori della funzione di probabilità
```

```
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileNoncentralF

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione-F non centrale inversa di Fisher con i parametri nu1, nu2 e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralF(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione-F non centrale inversa di Fisher con i parametri nu1, nu2 e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralF(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione distribuzione-F non centrale inversa di Fisher con i parametri nu1, nu2 e sigma. In caso di errore restituisce false. Analogo di qf() in R.

```
double  MathQuantileNoncentralF(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione distribuzione-F non centrale inversa di Fisher con i parametri nu1, nu2 e sigma. In caso di errore restituisce false.

```
bool  MathQuantileNoncentralF(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomNoncentralF

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma. In caso di errore restituisce NaN.

```
double  MathRandomNoncentralF(
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma. In caso di errore restituisce false. Analogo dirf() in R.

```
bool  MathRandomNoncentralF(
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsNoncentralF

Calcola i valori numerici teorici dei primi 4 momenti di distribuzione-F non centrale di Fisher con i parametri nu1, nu2 e sigma.

```
double  MathMomentsNoncentralF(
   const double  nu1,            // il primo parametro della distribuzione (numero di
   const double  nu2,            // il secondo parametro della distribuzione (numero d
   const double  sigma,          // parametro noncentralità
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

nu1

[in] Il primo parametro della distribuzione (numero di gradi di libertà).

nu2

[in] Il secondo parametro della distribuzione (numero di gradi di libertà).

sigma

[in] Parametro Noncentrality.

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

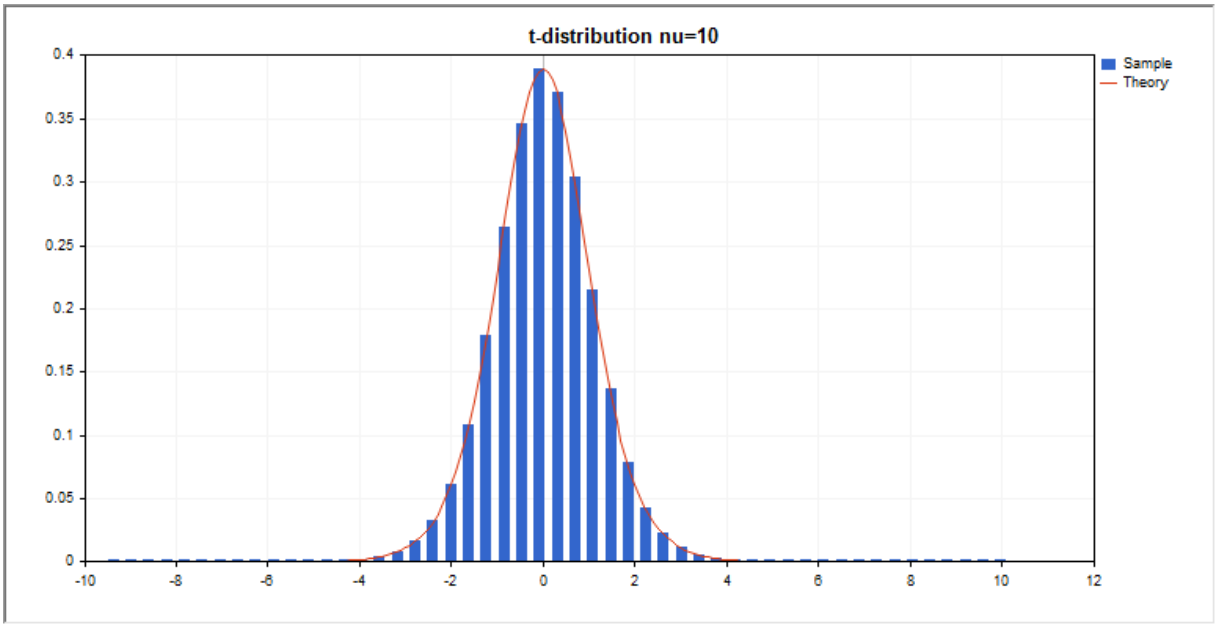
Distribuzione-t

Questa sezione contiene le funzioni per lavorare con distribuzione-t di Student. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge dello studente. La distribuzione-t è definita dalla seguente formula:

$$f_T(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

dove:

- x — valore della variabile casuale
- ν — il parametro della distribuzione (numero di gradi di libertà)



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityT	Calcola la funzione di densità di probabilità della distribuzione-t
MathCumulativeDistributionT	Calcola il valore della funzione di distribuzione-t
MathQuantileT	Calcola il valore della funzione inversa della distribuzione-t per la probabilità specificata
MathRandomT	Genera una variabile pseudocasuale / array di variabili pseudocasuali distribuiti secondo la legge di distribuzione-t di Student

Funzione	Descrizione
MathMomentsT	Calcola i valori numerici teoriche dei primi 4 momenti della distribuzione-t di Student

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\T.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double nu_par=10;     // il numero di gradi di libertà
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // il numero di valori nell'esempio
   int ncells=51;        // il numero di intervalli nell'istogramma
   double x[];           // centro degli intervalli dell'istogramma
   double y[];           // il numero di valori dall'esempio che cade all'interno dell
   double data[];        // esempio di valori casuali
   double max,min;       // i valori massimo e minimo nell'esempio
//--- ottiene un campione da distribuzione-t di Student
   MathRandomT(nu_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityT(x2,nu_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
```

```
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("t-distribution nu=%G",nu_par));
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityT

Calcola il valore della funzione di densità di probabilità di distribuzione-t di Student con il parametro nu per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-t di Student con il parametro nu per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-t di Student con il parametro nu per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dt() in R.

```
bool  MathProbabilityDensityT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-t di Student con il parametro nu per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionT

Calcola il valore della funzione di distribuzione-t di Student con il parametro nu per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione-t di Student con il parametro nu per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione-t di Student con il parametro nu per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pt() in R.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione-t di Student con il parametro nu per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileT

Per la specificata *probabilità*, la funzione calcola il valore della funzione distribuzione t di student inversa con il parametro nu. In caso di errore restituisce NaN.

```
double  MathQuantileT(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione distribuzione t di student inversa con il parametro nu. In caso di errore restituisce NaN.

```
double  MathQuantileT(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione distribuzione-t di Student inversa con il parametro nu. In caso di errore restituisce false. Analogo di qt() in R.

```
double  MathQuantileT(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vien
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione distribuzione-t di Student inversa con il parametro nu. In caso di errore restituisce false.

```
bool  MathQuantileT(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  nu,             // parametro della distribuzione (numero di gradi di
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomT

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione-t di Student con il parametro nu. In caso di errore restituisce NaN.

```
double  MathRandomT(
   const double  nu,              // parametro della distribuzione (numero di gradi di
   int&          error_code       // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione-t di Student con il parametro nu. In caso di errore restituisce false. Analogo di rt() in R.

```
bool  MathRandomT(
   const double  nu,              // parametro della distribuzione (numero di gradi di
   const int     data_count,      // ammontare dei dati richiesti
   double&       result[]         // array con i valori delle variabili pseudocasuali
   );
```

Parametri

nu

[in] Parametro di distribuzione (numero di gradi di libertà).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsT

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione-t di Student con il parametro nu.

```
double  MathMomentsT(
   const double  nu,              // parametro della distribuzione (numero di gradi di
   double&       mean,            // variabile per la media
   double&       variance,        // variabile per la varianza
   double&       skewness,        // variabile per l'asimmetria
   double&       kurtosis,        // variabile per la curtosi
   int&          error_code       // variabile per il codice errore
   );
```

Parametri

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

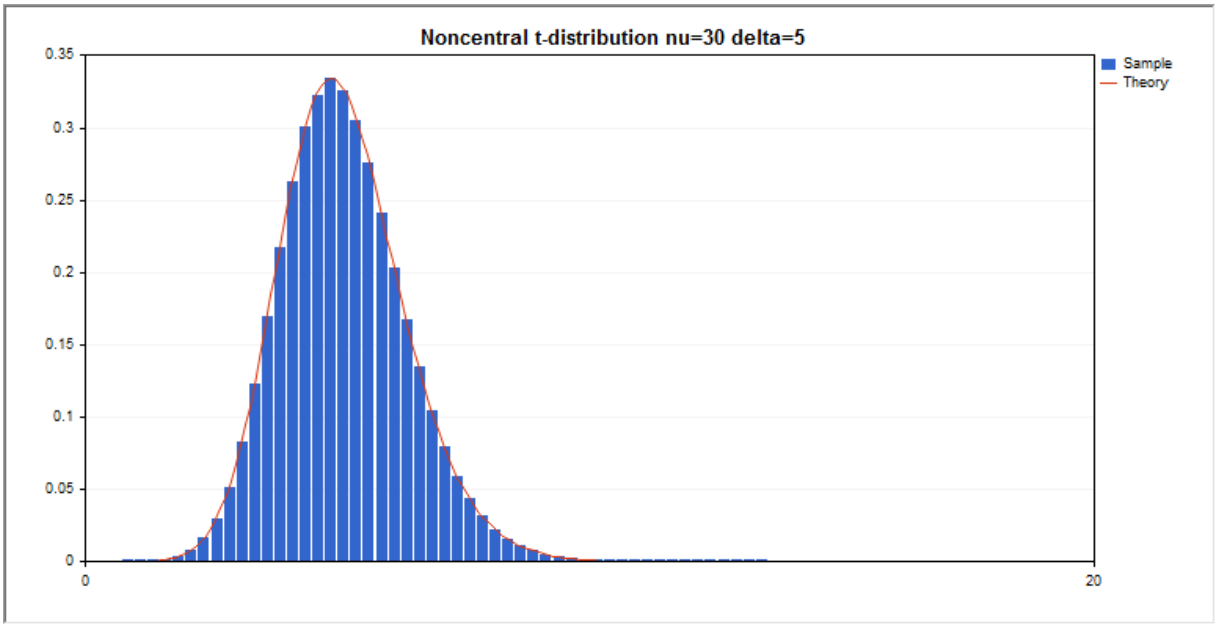
Distribuzione-t

Questa sezione contiene le funzioni per lavorare con distribuzione-t non centrale di Student. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge di distribuzione-t non centrale. La distribuzione-t non centrale è definita dalla seguente formula:

$$f_{NoncentralT}(x \mid \nu, \delta) = \frac{\nu^{\frac{\nu}{2}} e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu+x^2\right)^{\frac{\nu+1}{2}}} \sum_{r=0}^{\infty} \frac{(x\delta)^r}{r!}\left(\frac{2}{\nu+x^2}\right)^{\frac{r}{2}} \Gamma\left(\frac{\nu+r+1}{2}\right)$$

dove:

- x — valore della variabile casuale
- ν — il parametro della distribuzione (numero di gradi di libertà)
- σ — parametro non centralità



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityNoncentralT	Calcola la funzione di densità di probabilità della distribuzione-t non centrale
MathCumulativeDistributionNoncentralT	Calcola il valore della funzione di distribuzione-t non centrale
MathQuantileNoncentralT	Calcola il valore della funzione di distribuzione-t non centrale inversa per la probabilità specificata
MathRandomNoncentralT	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione-t non centrale di Student

Funzione	Descrizione
MathMomentsNoncentralT	Calcola i valori numerici teoriche dei primi 4 momenti della distribuzione-t non centrale di Student

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralT.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double nu_par=30;        // il numero di gradi di libertà
input double delta_par=5;     // parametro non centralità
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // il numero di valori nell'esempio
   int ncells=51;        // il numero di intervalli nell'istogramma
   double x[];           // centro degli intervalli dell'istogramma
   double y[];           // il numero di valori dall'esempio che cade all'interno dell
   double data[];        // esempio di valori casuali
   double max,min;       // i valori massimo e minimo nell'esempio
//--- ottiene un campione da distribuzione- t non centrale di Student
   MathRandomNoncentralT(nu_par,delta_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralT(x2,nu_par,delta_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral t-distribution nu=%G delta=%G",nu_p
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

```
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralT

Calcola il valore della funzione di densità di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  delta,          // paramtero noncentralità
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  delta,          // paramtero noncentralità
   int&          error_code      // variabile per il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dt() in R.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

delta

[in] Parametro Noncentrality.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionNoncentralT

Calcola la funzione di distribuzione di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  delta,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // valore di variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di l
   const double  delta,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pt() in R.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità di distribuzione-t non centrale di Student con i parametri nu e delta per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // array con i valori della variabile random
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

delta

[in] Parametro Noncentrality.

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileNoncentralT

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione-t non centrale di Student inversa con i parametri nu e delta. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se lower_tail=false, allora il ca
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione-t non centrale di Student inversa con i parametri nu e delta. In caso di errore restituisce NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione distribuzione-t non centrale di Student inversa con i parametri nu e delta. In caso di errore restituisce false. Analogo di qt() in R.

```
double  MathQuantileNoncentralT(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   const bool    tail,           // flag di calcolo, se lower_tail=false, allora il ca
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione distribuzione-t non centrale di Student inversa con i parametri nu e delta. In caso di errore restituisce false.

```
bool  MathQuantileNoncentralT(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

nu

[in] Parametro della distribuzione (numero di gradi di libertà).

delta

[in] Parametro Noncentrality.

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomNoncentralT

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione-t non centrale di Student con i parametri nu e delta. In caso di errore restituisce NaN.

```
double  MathRandomNoncentralT(
   const double  nu,              // parametro della distribuzione (numero di gradi di
   const double  delta,           // parametro noncentralità
   int&          error_code       // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione-t non centrale di Student con i parametri nu e delta. In caso di errore restituisce false. Analogo di rt() in R.

```
bool  MathRandomNoncentralT(
   const double  nu,              // parametro della distribuzione (numero di gradi di
   const double  delta,           // parametro noncentralità
   const int     data_count,      // ammontare dei dati richiesti
   double&       result[]         // array con i valori delle variabili pseudocasuali
   );
```

Parametri

nu

[in] Parametro di distribuzione (numero di gradi di libertà).

delta

[in] Parametro Noncentrality.

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsNoncentralT

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione-t non centrale di Student con i parametri nu e delta.

```
double  MathMomentsNoncentralT(
   const double  nu,             // parametro della distribuzione (numero di gradi di
   const double  delta,          // parametro noncentralità
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

nu

[in] Parametro di distribuzione (numero di gradi di libertà).

delta

[in] Parametro noncentralità.

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

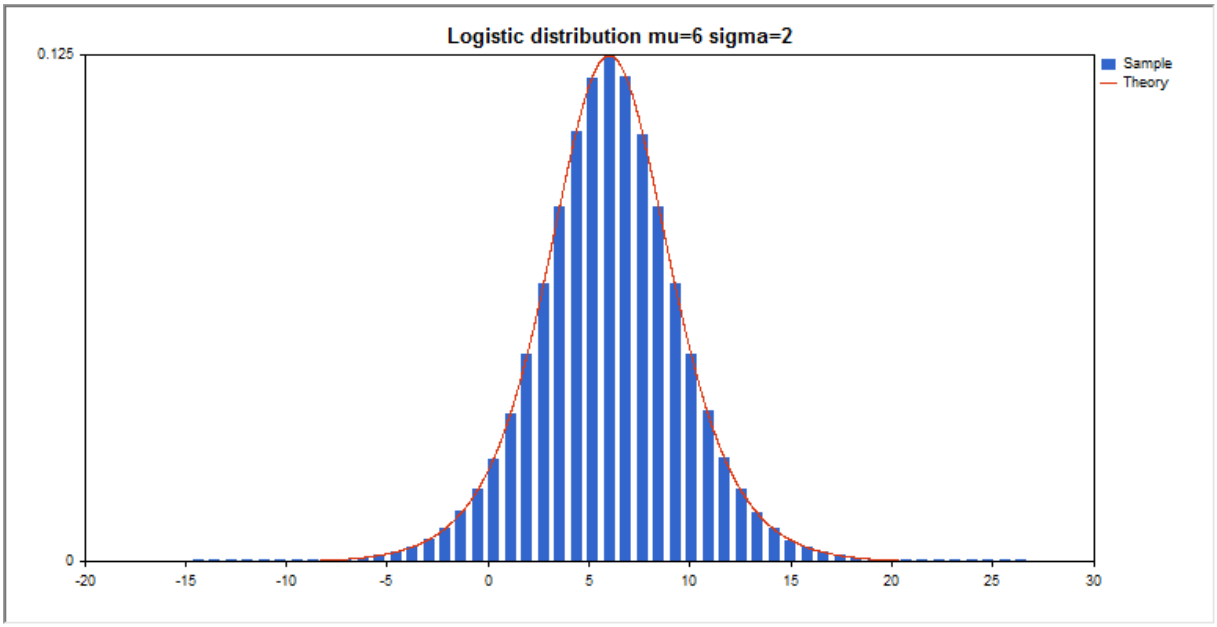
Distribuzione logistica

Questa sezione contiene funzioni per lavorare con distribuzione logistica. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge logistica. Distribuzione logistica è definita dalla seguente formula:

$$f_{Logistic}(x|\mu,\sigma) = \frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

dove:

- x — valore della variabile casuale
- μ — parametro medio della distribuzione
- Σ — parametro di scala della distribuzione



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityLogistic	Calcola la funzione di densità di probabilità della distribuzione logistica
MathCumulativeDistributionLogistic	Calcola il valore della funzione di distribuzione di probabilità logistica
MathQuantileLogistic	Calcola il valore della funzione di distribuzione logistica inversa per la probabilità specificata
MathRandomLogistic	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge logistica

Funzione	Descrizione
MathMomentsLogistic	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione logistica

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Logistic.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double mu_par=6;        // parametro medio della distribuzione
input double sigma_par=2;     // parametro scala della distribuzione
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // il numero di valori nell'esempio
   int ncells=51;       // il numero di intervalli nell'istogramma
   double x[];          // centro degli intervalli dell'istogramma
   double y[];          // il numero di valori dall'esempio che cade all'interno dell
   double data[];       // esempio di valori casuali
   double max,min;      // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione logistica
   MathRandomLogistic(mu_par,sigma_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLogistic(x2,mu_par,sigma_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Logistic distribution mu=%G sigma=%G",mu_par,s
   graphic.BackgroundMainSize(16);
//--- disabilita la scalatura automatica dell'asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityLogistic

Calcola il valore della funzione di densità di probabilità di distribuzione logistica con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro medio della distribuzione
   const double  sigma,          // parametro scala della distribuzione
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione logistica con i parametri mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro medio della distribuzione
   const double  sigma,          // parametro scala della distribuzione
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione logistica con i parametri mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dlogis() in R.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità di distribuzione logistica con i parametri mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] parametro medio della distribuzione.

sigma

[in] parametro scala della distribuzione.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionLogistic

Calcola la funzione di distribuzione logistica delle probabilità con i parametri di mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro medio della distribuzione
   const double  sigma,          // parametro scala della distribuzione
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione logistica delle probabilità con i parametri di mu e sigma per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // valore di variabile random
   const double  mu,             // parametro medio della distribuzione
   const double  sigma,          // parametro scala della distribuzione
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione logistica delle probabilità con i parametri di mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione logistica delle probabilità con i parametri di mu e sigma per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di plogis() in R.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // array con i valori della variabile random
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

mu

[in] parametro medio della distribuzione.

sigma

[in] parametro scala della distribuzione.

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileLogistic

Per la specificata *probabilità*, la funzione calcola il valore della funzione inversa di distribuzione logistica con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione inversa di distribuzione logistica con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa di distribuzione logistica con i parametri mu e sigma. In caso di errore restituisce false. Analogo di qlogis() in R.

```
double  MathQuantileLogistic(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa di distribuzione logistica con i parametri mu e sigma. In caso di errore restituisce false.

```
bool  MathQuantileLogistic(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

mu

[in] parametro medio della distribuzione.

sigma

[in] parametro scala della distribuzione.

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomLogistic

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione logistica con i parametri mu e sigma. In caso di errore restituisce NaN.

```
double  MathRandomLogistic(
   const double  mu,              // parametro medio per la distribuzione
   const double  sigma,           // parametro scala per la distribuzione
   int&          error_code       // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione logistica con i parametri mu e sigma. In caso di errore restituisce false. Analogo di rlogis() in R.

```
bool  MathRandomLogistic(
   const double  mu,              // parametro medio per la distribuzione
   const double  sigma,           // parametro scala per la distribuzione
   const int     data_count,      // ammontare dei dati richiesti
   double&       result[]         // array con i valori delle variabili pseudocasuali
   );
```

Parametri

mu

[in] parametro medio della distribuzione.

sigma

[in] parametro scala della distribuzione.

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsLogistic

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione logistica con i parametri mu e sigma.

```
double  MathMomentsLogistic(
   const double  mu,             // parametro medio per la distribuzione
   const double  sigma,          // parametro scala per la distribuzione
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

mu

[in] parametro medio della distribuzione.

sigma

[in] parametro scala della distribuzione.

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

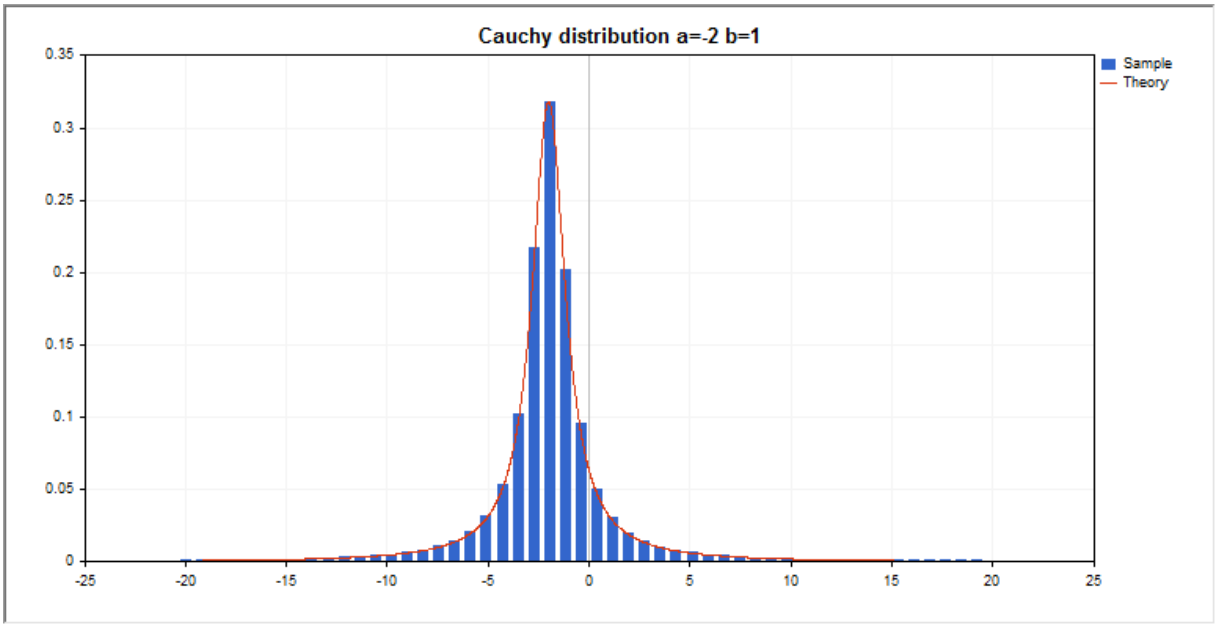
Distribuzione di Cauchy

Questa sezione contiene funzioni per lavorare con distribuzione di Cauchy. Esse permettono di calcolare la densità, probabilità, quantili e di generare numeri pseudo-casuali distribuiti secondo la legge di Cauchy. La distribuzione Cauchy è definita dalla seguente formula:

$$f_{Cauchy}(x|a,b) = \frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

dove:

- x — valore della variabile casuale
- a — media parametro della distribuzione
- b — parametro di scala della distribuzione



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityCauchy	Calcola la funzione di densità di probabilità della distribuzione di Cauchy
MathCumulativeDistributionCauchy	Calcola il valore della funzione di distribuzione di probabilità di Cauchy
MathQuantileCauchy	Calcola il valore della funzione di distribuzione di Cauchy inversa per la probabilità specificata
MathRandomCauchy	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge Cauchy
MathMomentsCauchy	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione Cauchy

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Cauchy.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double a_par=-2;      // parametro media della distribuzione
input double b_par=1;       // parametro scala della distribuzione
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // il numero di valori nell'esempio
   int ncells=51;          // il numero di intervalli nell'istogramma
   double x[];             // centro degli intervalli dell'istogramma
   double y[];             // il numero di valori dall'esempio che cade all'interno dell
   double data[];          // esempio di valori casuali
   double max,min;         // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione Cauchy
   MathRandomCauchy(a_par,b_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityCauchy(x2,a_par,b_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Cauchy distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1)
      return(false);
   int size=ArraySize(data);
   if(size<cells*10)
      return(false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   Print("min=",minv," max=",maxv);
   minv=-20;
   maxv=20;
   double range=maxv-minv;
   double width=range/cells;
   if(width==0)
      return(false);
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=(int)MathRound((data[i]-minv)/width);
      if(ind>=0 && ind<cells)
         frequency[ind]++;
     }
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityCauchy

Calcola il valore della funzione di densità di probabilità della distribuzione Cauchy con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // valore di variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione Cauchy con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // valore di variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione Cauchy con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dcauchy() in R.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione Cauchy con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] parametro medio della distribuzione.

b

[in] parametro scala della distribuzione.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionCauchy

Calcola la funzione di distribuzione di probabilità della distribuzione di Cauchy con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // valore di variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità della distribuzione di Cauchy con i parametri A e B per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // valore di variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione della probabilità della distribuzione di Cauchy con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità della distribuzione di Cauchy con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analog of the pcauchy() in R.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] parametro medio della distribuzione.

b

[in] parametro scala della distribuzione.

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileCauchy

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione Cauchy inversa con parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione Cauchy inversa con parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione Cauchy inversa con parametri A e B. In caso di errore restituisce false. Analogo di qcauschy() in R.

```
double  MathQuantileCauchy(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione Cauchy inversa con parametri A e B. In caso di errore restituisce false.

```
bool  MathQuantileCauchy(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

a

[in] parametro medio della distribuzione.

b

[in] parametro scala della distribuzione.

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomCauchy

Genera una variabile pseudocasuale distribuita in base alla legge di distribuzione di Cauchy con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathRandomCauchy(
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge della distribuzione di Cauchy con i parametri A e B. In caso di errore restituisce false. Analogo di rcauchy() in R.

```
bool  MathRandomCauchy(
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

a

[in] parametro medio della distribuzione.

b

[in] parametro scala della distribuzione.

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsCauchy

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione Cauchy con i parametri A e B.

```
double  MathMomentsCauchy(
   const double  a,              // parametro medio della distribuzione
   const double  b,              // parametro scala della distribuzione
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

a

[in] parametro medio della distribuzione.

b

[in] parametro scala della distribuzione.

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

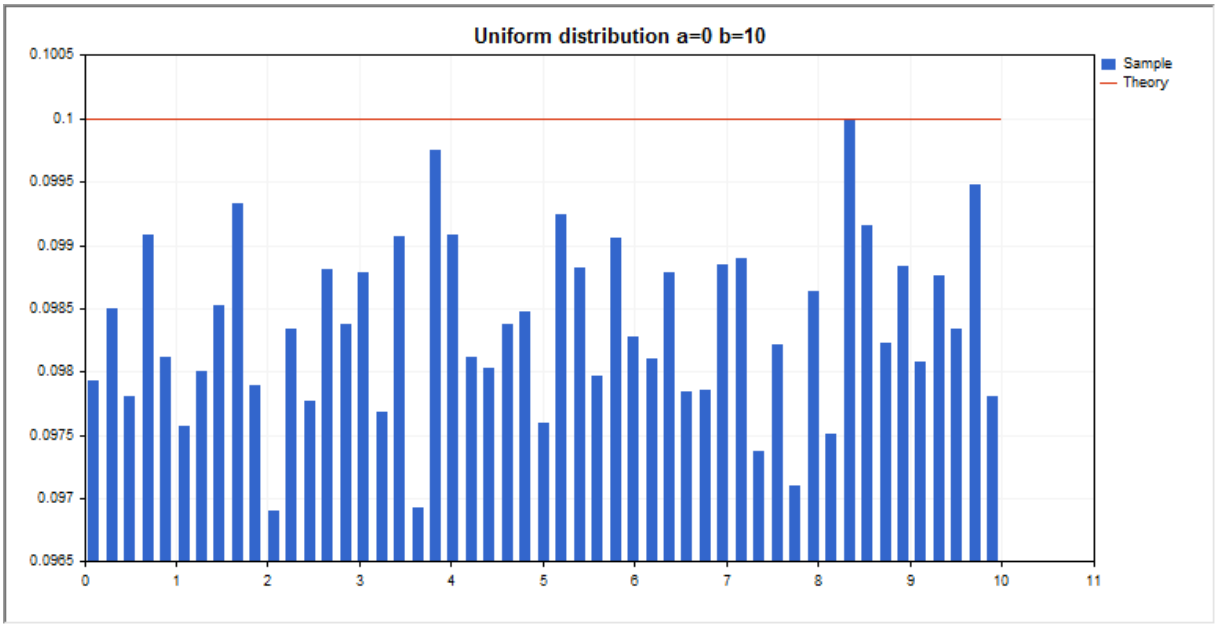
Distribuzione uniforme

Questa sezione contiene funzioni per lavorare con la distribuzione uniforme. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge uniforme. La distribuzione uniforme è definita dalla seguente formula:

$$f_{Uniform}(x|\ a,b) = \frac{1}{b-a}$$

dove:

- x — valore della variabile casuale
- a — parametro della distribuzione (limite inferiore)
- b — parametro della distribuzione (limite superiore)



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityUniform	Calcola la funzione di densità di probabilità della distribuzione uniforme
MathCumulativeDistributionUniform	Calcola il valore della funzione di distribuzione di probabilità uniforme
MathQuantileUniform	Calcola il valore della funzione di distribuzione uniforme inversa per la probabilità specificata
MathRandomUniform	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge uniforme
MathMomentsUniform	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione uniforme

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Uniform.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double a_par=0;      // parametro distribuzione a (confine inferiore)
input double b_par=10;     // parametro distribuzione b (confine superiore)
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // il numero di valori nell'esempio
   int ncells=51;          // il numero di intervalli nell'istogramma
   double x[];             // centro degli intervalli dell'istogramma
   double y[];             // il numero di valori dall'esempio che cade all'interno dell
   double data[];          // esempio di valori casuali
   double max,min;         // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla una distribuzione uniforme
   MathRandomUniform(a_par,b_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityUniform(x2,a_par,b_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Uniform distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityUniform

Calcola il valore della funzione di densità di probabilità della distribuzione uniforme con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,              // valore di variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione uniforme con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,              // valore di variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione uniforme con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dunif() in R.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione uniforme con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Parametro della distribuzione a (limite inferiore).

b

[in] Parametro della distribuzione b (limite superiore).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionUniform

Calcola la funzione di distribuzione di probabilità di una distribuzione uniforme con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // valore di variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di una distribuzione uniforme con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // valore di variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola la funzione di distribuzione di probabilità di una distribuzione uniforme con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola la funzione di distribuzione di probabilità di una distribuzione uniforme con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di punif() in R.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Parametro della distribuzione a (limite inferiore).

b

[in] Parametro della distribuzione b (limite superiore).

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileUniform

Per la specificata *probabilità*, la funzione calcola il valore della funzione inversa distribuzione uniforme con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileUniform(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione inversa distribuzione uniforme con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileUniform(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione uniforme con i parametri A e B. In caso di errore restituisce false. Analogo di qunif() in R.

```
double  MathQuantileUniform(
   const double& probability[],  // array con i valori della proabilità della variabi
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione inversa distribuzione uniforme con i parametri A e B. In caso di errore restituisce false.

```
bool  MathQuantileUniform(
   const double& probability[],  // array i valori della probabilità della variabile
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

a

[in] Parametro della distribuzione a (limite inferiore).

b

[in] Parametro della distribuzione b (limite superiore).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomUniform

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione uniforme con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathRandomUniform(
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuiti secondo la legge di distribuzione uniforme con i parametri A e B. In caso di errore restituisce false. Analogo di runif() in R.

```
bool  MathRandomUniform(
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

a

[in] Parametro della distribuzione a (limite inferiore).

b

[in] Parametro della distribuzione b (limite superiore).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsUniform

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione uniforme con i parametri A e B.

```
double  MathMomentsUniform(
   const double  a,              // parametro della distribuzione a (limite inferiore)
   const double  b,              // parametro della distribuzione b (limite superiore)
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

a

[in] Parametro della distribuzione a (limite inferiore).

b

[in] Parametro della distribuzione b (limite superiore).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

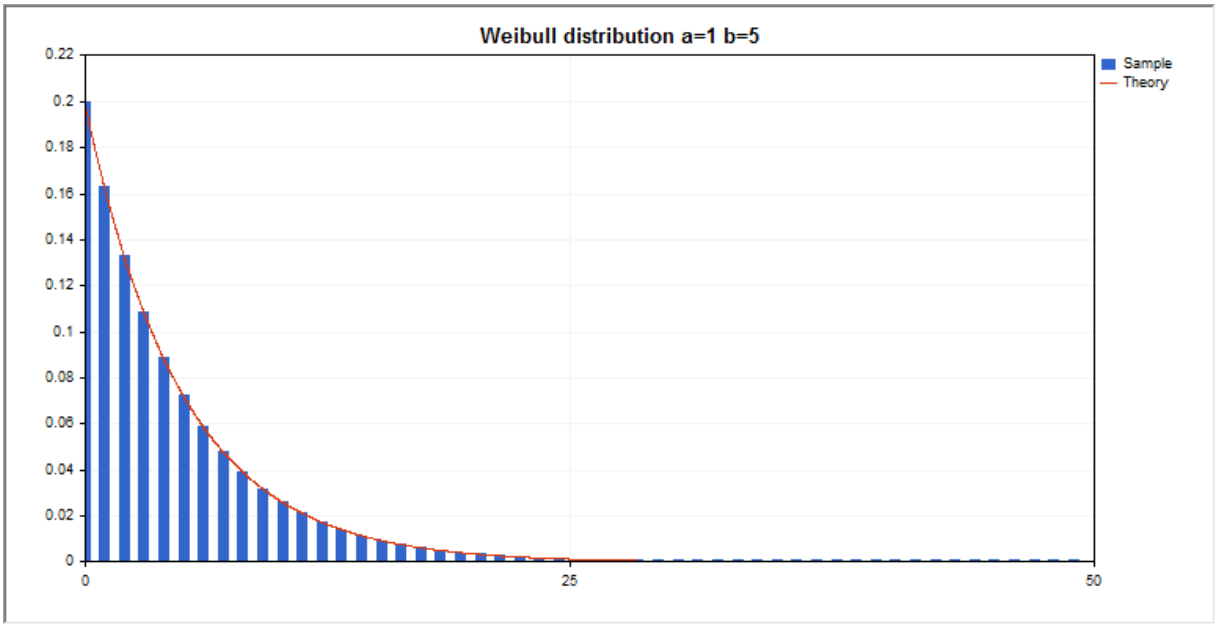
Distribuzione di Weibull

Questa sezione contiene funzioni per lavorare con la distribuzione Weibull. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge Weibull. La distribuzione Weibull è definita dalla seguente formula:

$$f_{Weibull}(x|a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

dove:

- x — valore della variabile casuale
- a — parametro della distribuzione (forma)
- b — parametro della distribuzione (scala)



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityWeibull	Calcola la funzione di densità di probabilità della distribuzione di Weibull
MathCumulativeDistributionWeibull	Calcola il valore della funzione di distribuzione di probabilità di Weibull
MathQuantileWeibull	Calcola il valore della funzione di distribuzione di Weibull inversa per la probabilità specificata
MathRandomWeibull	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di Weibull
MathMomentsWeibull	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione Weibull

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Weibull.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double a_par=1;      // parametro della distribuzione (shape)
input double b_par=5;      // parametro della distribuzione (scale)
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // il numero di valori nell'esempio
   int ncells=51;          // il numero di intervalli nell'istogramma
   double x[];             // centro degli intervalli dell'istogramma
   double y[];             // il numero di valori dall'esempio che cade all'interno dell
   double data[];          // esempio di valori casuali
   double max,min;         // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione Weibull
   MathRandomWeibull(a_par,b_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityWeibull(x2,a_par,b_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Weibull distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- disabilitare scalatura automatica dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityWeibull

Calcola il valore della funzione di densità di probabilità della distribuzione Weibull con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // valore di variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione Weibull con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // valore di variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione Weibull con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dweibull() in R.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di densità di probabilità della distribuzione Weibull con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Parametro della distribuzione (scala).

b

[in] Parametro della distribuzione (forma).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionWeibull

Calcola il valore della funzione di distribuzione Weibull con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // valore di variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione Weibull con i parametri A e B per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // valore di variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione Weibull con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pweibull() in R.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione Weibull con i parametri A e B per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // array con i valori della variabile random
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

a

[in] Parametro della distribuzione (scala).

b

[in] Parametro della distribuzione (forma).

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileWeibull

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione Weibull inversa con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore della funzione di distribuzione Weibull inversa con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione Weibull inversa con parametri A e B. In caso di errore restituisce false. Analogo di qweibull() in R.

```
double  MathQuantileWeibull(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore della funzione di distribuzione Weibull inversa con parametri A e B. In caso di errore restituisce false.

```
bool  MathQuantileWeibull(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

a

[in] Parametro della distribuzione (scala).

b

[in] Parametro della distribuzione (forma).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomWeibull

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione di Weibull con i parametri A e B. In caso di errore restituisce NaN.

```
double  MathRandomWeibull(
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuiti secondo la legge di distribuzione di Weibull con i parametri A e B. In caso di errore restituisce false. Analogo di rweibull() in R.

```
bool  MathRandomWeibull(
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

a

[in] Parametro della distribuzione (scala).

b

[in] Parametro della distribuzione (forma).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsWeibull

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione Weibull con i parametri A e B.

```
double  MathMomentsWeibull(
   const double  a,              // primo parametro della distribuzione (forma)
   const double  b,              // primo parametro della distribuzione (scala)
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

a

[in] Parametro della distribuzione (scala).

b

[in] Parametro della distribuzione (forma).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

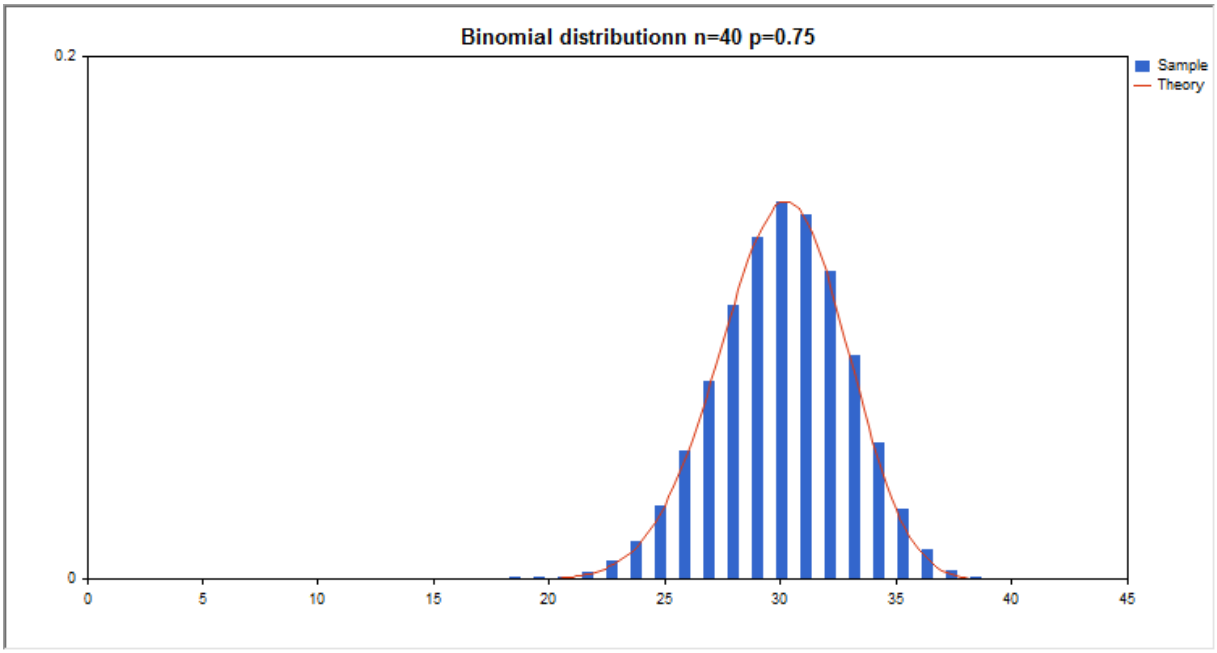
Distribuzione binomiale

Questa sezione contiene funzioni per lavorare con la distribuzione binomiale. Esse permettono di calcolare la densità, probabilità, quantili e di generare numeri pseudo-casuali distribuiti secondo la legge binomiale. La distribuzione binomiale è definita dalla seguente formula:

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

dove:

- x — valore della variabile casuale
- n — numero di test
- p — probabilità di successo per ogni test



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityBinomial	Calcola la funzione di densità di probabilità della distribuzione binomiale
MathCumulativeDistributionBinomial	Calcola il valore della funzione di distribuzione di probabilità binomiale
MathQuantileBinomial	Calcola il valore della funzione di distribuzione binomiale inversa per la probabilità specificata
MathRandomBinomial	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge binomiale
MathMomentsBinomial	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione binomiale

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Binomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double n_par=40;         // il numero di tests
input double p_par=0.75;       // probabilità di successo per ogni test
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // il numero di valori nell'esempio
   int ncells=20;          // il numero di intervalli nell'istogramma
   double x[];             // centro degli intervalli dell'istogramma
   double y[];             // il numero di valori dall'esempio che cade all'interno dell
   double data[];          // esempio di valori casuali
   double max,min;         // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione binomiale
   MathRandomBinomial(n_par,p_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityBinomial(x2,n_par,p_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Binomial distributionn n=%G p=%G",n_par,p_par)
```

```
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

MathProbabilityDensityBinomial

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale con i parametri N e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valore di variabile random
   const double  n,              // parametri per la distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale con i parametri N e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valore di variabile random
   const double  n,              // parametri per la distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale con i parametri N e P per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dbinom() in R.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occo
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se lo
   double&       result[]        // array per i valori della funzione di densita di p
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale con i parametri N e P per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occo
   double&       result[]        // array per i valori della funzione di densita di p
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

n

[in] Parametro della distribuzione (numero di test).

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionBinomial

Calcola il valore della funzione di distribuzione di probabilità per legge binomiale con i parametri N e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // valore di variabile random
   const double  n,              // parametri per la distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge binomiale con i parametri N e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // valore di variabile random
   const double  n,              // parametri per la distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per la legge binomiale con i parametri N e P per una serie di variabili casuali x[]. In caso di errore restituisce false. Analogo di pbinom() in R.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occo
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge binomiale con i parametri N e P per una serie di variabili casuali x []. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occo
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

n

[in] Parametro della distribuzione (numero di test).

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

tail

[in] Flag di calcolo. Se true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileBinomial

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per la legge binomiale con i parametri N e P. In caso di errore restituisce NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per la legge binomiale con i parametri N e P. In caso di errore restituisce NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per la legge binomiale con i parametri N e P. In caso di errore restituisce false. Analogo di qbinom() in R.

```
double  MathQuantileBinomial(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per la legge binomiale con i parametri N e P. In caso di errore restituisce false.

```
bool  MathQuantileBinomial(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occor
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

n

[in] Parametro della distribuzione (numero di test).

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomBinomial

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione binomiale con i parametri N e P. In caso di errore restituisce NaN.

```
double  MathRandomBinomial(
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occo
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuiti secondo la legge di distribuzione binomiale con i parametri N e P. In caso di errore restituisce false. Analogo di rbinom() in R.

```
bool  MathRandomBinomial(
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occo
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

n

[in] Parametro della distribuzione (numero di test).

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsBinomial

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione binomiale con parametri N e P.

```
double  MathMomentsBinomial(
   const double  n,              // parametro della distribuzione (numero di tests)
   const double  p,              // parametro della distribuzione (probabilità o occo
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

n

[in] Parametro della distribuzione (numero di test).

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

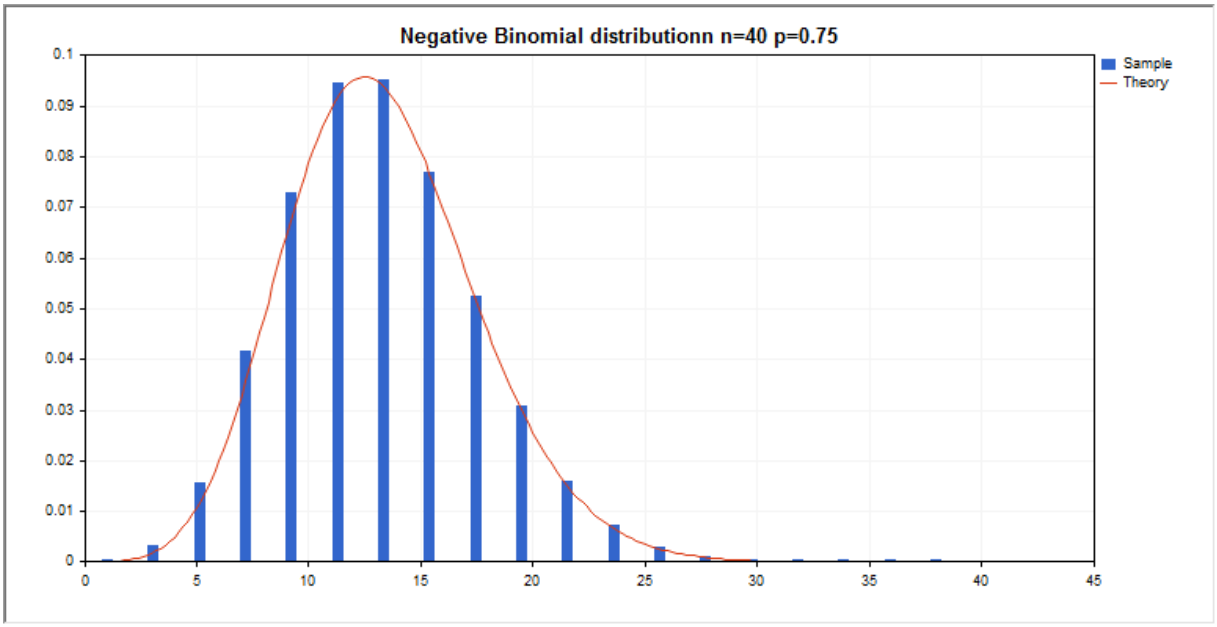
Distribuzione binomiale negativa

Questa sezione contiene funzioni per lavorare con la distribuzione binomiale negativa. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge binomiale negativa. La distribuzione binomiale negativa è definita dalla seguente formula:

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

dove:

- x — valore della variabile casuale
- r — numero di test di successo
- p — probabilità di successo



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityNegativeBinomial	Calcola la funzione di densità di probabilità della distribuzione binomiale negativa
MathCumulativeDistributionNegativeBinomial	Calcola il valore della funzione di distribuzione di probabilità binomiale negativa
MathQuantileNegativeBinomial	Calcola il valore della funzione di distribuzione binomiale negativa inversa per la probabilità specificata
MathRandomNegativeBinomial	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge binomiale negativa
MathMomentsNegativeBinomial	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione binomiale negativa

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NegativeBinomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double n_par=40;         // il numero di tests
input double p_par=0.75;       // probabilità di successo per ogni test
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;         // il numero di valori nell'esempio
   int ncells=19;         // il numero di intevalli nell'istogramma
   double x[];            // centro degli intervalli dell'istogramma
   double y[];            // il numero di valori dall'esempio che cade all'interno dell
   double data[];         // esempio di valori casuali
   double max,min;        // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione binomiale negativa
   MathRandomNegativeBinomial(n_par,p_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityNegativeBinomial(x2,n_par,p_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Negative Binomial distributionn n=%G p=%G",n_p
```

```
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

MathProbabilityDensityNegativeBinomial

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale negativa con parametri R e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // valore della variabile random (integer)
   const double  r,              // numero di test con successo
   const double  p,              // probabilità di successo
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale negativa con parametri R e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // valore della variabile random (integer)
   const double  r,              // numero di test con successo
   const double  p,              // probabilità di successo
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale negativa con parametri R e P per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dnbinom() in R.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione binomiale negativa con parametri R e P per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

r

[in] Numero di test di successo

p

[in] Probabilità di successo.

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionNegativeBinomial

Calcola il valore della funzione di distribuzione di probabilità per la legge binomiale con i parametri R e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // valore della variabile random (integer)
   const double  r,              // numero di test con successo
   const double  p,              // probabilità di successo
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per la legge binomiale con i parametri R e P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // valore della variabile random (integer)
   const double  r,              // numero di test con successo
   const double  p,              // probabilità di successo
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per la legge binomiale con i parametri R e P per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di pnbinom() in R.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione di probabilità per la legge binomiale con i parametri R e P per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // array con i valori della variabile random
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

r

[in] Numero di tests di successo.

p

[in] Probabilità di successo.

tail

[in] Flag di calcolo, se è true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore, se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileNegativeBinomial

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per legge binomiale negativa con i parametri R e P. In caso di errore restituisce NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per legge binomiale negativa con i parametri R e P. In caso di errore restituisce NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per la legge binomiale con i parametri R e P. In caso di errore restituisce false. Analogo di qnbinom() in R.

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per la legge binomiale con i parametri R e P. In caso di errore restituisce false.

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

r

[in] Numero di tests di successo.

p

[in] Probabilità di successo.

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomNegativeBinomial

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione binomiale con i parametri N e P. In caso di errore restituisce NaN.

```
double  MathRandomNegativeBinomial(
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuiti secondo la legge di distribuzione binomiale con i parametri N e P. In caso di errore restituisce false. Analogo di rnbinom() in R.

```
bool  MathRandomNegativeBinomial(
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

r

[in] Numero di tests di successo.

p

[in] Probabilità di successo.

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsNegativeBinomial

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione binomiale negativa con parametri R e P.

```
double  MathMomentsNegativeBinomial(
   const double  r,              // numero di test di successo
   const double  p,              // probabilità di successo
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

r

[in] Numero di tests di successo.

p

[in] Probabilità di successo.

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

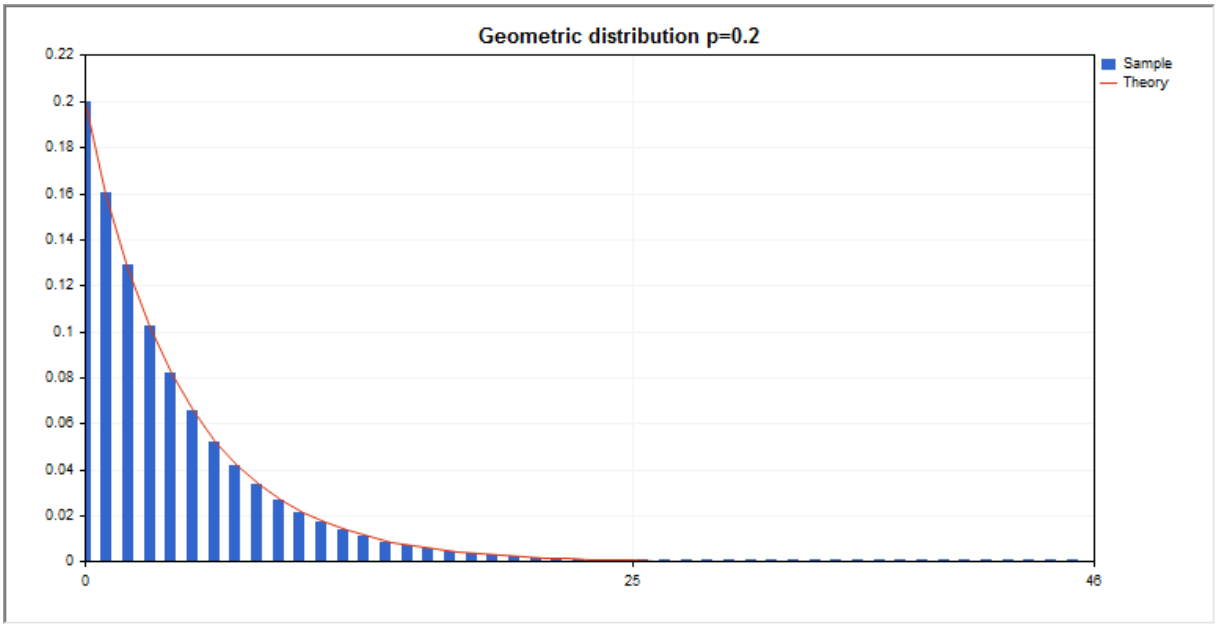
Distribuzione geometrica

Questa sezione contiene funzioni per lavorare con la distribuzione geometrica. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge geometrica. La distribuzione di geometrica è definita dalla seguente formula:

$$f_{Geometric}(x|\ p) = p(1-p)^x$$

dove:

- x — valore della variabile casuale (intero)
- p — probabilità di accadimento dell'evento in una prova



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityGeometric	Calcola la funzione di densità di probabilità della distribuzione geometrica
MathCumulativeDistributionGeometric	Calcola il valore della funzione di distribuzione di probabilità geometrica
MathQuantileGeometric	Calcola il valore della funzione di distribuzione geometrica inversa per la probabilità specificata
MathRandomGeometric	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione geometrica
MathMomentsGeometric	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione geometrica

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Geometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double p_par=0.2;      // probabilità di occorrenza di un evento in un test
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // il numero di valori nell'esempio
   int ncells=47;       // il numero di intervalli nell'istogramma
   double x[];          // centro degli intervalli dell'istogramma
   double y[];          // il numero di valori dall'esempio che cade all'interno dell
   double data[];       // esempio di valori casuali
   double max,min;      // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione geometrica
   MathRandomGeometric(p_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(0,ncells,1,x2);
   MathProbabilityDensityGeometric(x2,p_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

```
   graphic.BackgroundMain(StringFormat("Geometric distribution p=%G",p_par));
   graphic.BackgroundMainSize(16);
//--- disabilitare scalatura automatica dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
```

```
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityGeometric

Calcola il valore della funzione di massa di probabilità di distribuzione geometrica con parametro P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // valore della variabile random (integer)
   const double  p,              // parametro della distribuzione (probabilità o occori
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione geometrica con parametro P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // valore della variabile random (integer)
   const double  p,              // parametro della distribuzione (probabilità o occori
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione geometrica con parametro P per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo didgeom() in R.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // array con i valori della variabile random
   const double  p,              // parametro della distribuzione (probabilità o occor
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione geometrica con parametro P per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // array con i valori della variabile random
   const double  p,              // parametro della distribuzione (probabilità o occor
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionGeometric

Calcola il valore della funzione di distribuzione di probabilità per legge geometrica con parametro P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // valore della variabile random (integer)
   const double  p,              // parametro della distribuzione (probabilità o occor
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge geometrica con parametro P per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // valore della variabile random (integer)
   const double  p,              // parametro della distribuzione (probabilità o occor
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge geometrica con il parametro P per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo dipgeom() in R.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // array con i valori della variabile random
   const double  p,              // parametro della distribuzione (probabilità o occo
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge geometrica con il parametro P per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // array con i valori della variabile random
   const double  p,              // parametro della distribuzione (probabilità o occo
   double&       result[]        // array per i valori della funzione di probabilità
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

tail

[in] Flag di calcolo, se è true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore, se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di probabilità.

MathQuantileGeometric

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per legge geometrica con parametro P. In caso di errore restituisce NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  p,              // parametro della distribuzione (probabilità o occo
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vie
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per legge geometrica con parametro P. In caso di errore restituisce NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  p,              // parametro della distribuzione (probabilità o occo
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probabilità[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per legge geometrica con parametro P. In caso di errore restituisce false. Analogo di qgeom() in R.

```
double  MathQuantileGeometric(
   const double& probability[],  // array con i valori della proabilità della variabi
   const double  p,              // parametro della distribuzione (probabilità o occo
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vie
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probabilità[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per legge geometrica con parametro P. In caso di errore restituisce false.

```
bool  MathQuantileGeometric(
   const double& probability[],  // array i valori della probabilità della variabile
   const double  p,              // parametro della distribuzione (probabilità o occo
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

tail

[in] Flag del calcolo, se false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomGeometric

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione geometrica con parametro P. In caso di errore restituisce NaN.

```
double  MathRandomGeometric(
   const double  p,              // parametro della distribuzione (probabilità o occo
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione geometrica con parametro P. In caso di errore restituisce false. Analogo di rgeom() in R.

```
bool  MathRandomGeometric(
   const double  p,              // parametro della distribuzione (probabilità o occo
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsGeometric

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione geometrica con parametro P.

```
double  MathMomentsGeometric(
   const double  p,              // parametro della distribuzione (probabilità o occo
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

p

[in] parametro della distribuzione (probabilità o il verificarsi dell'evento in un test).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

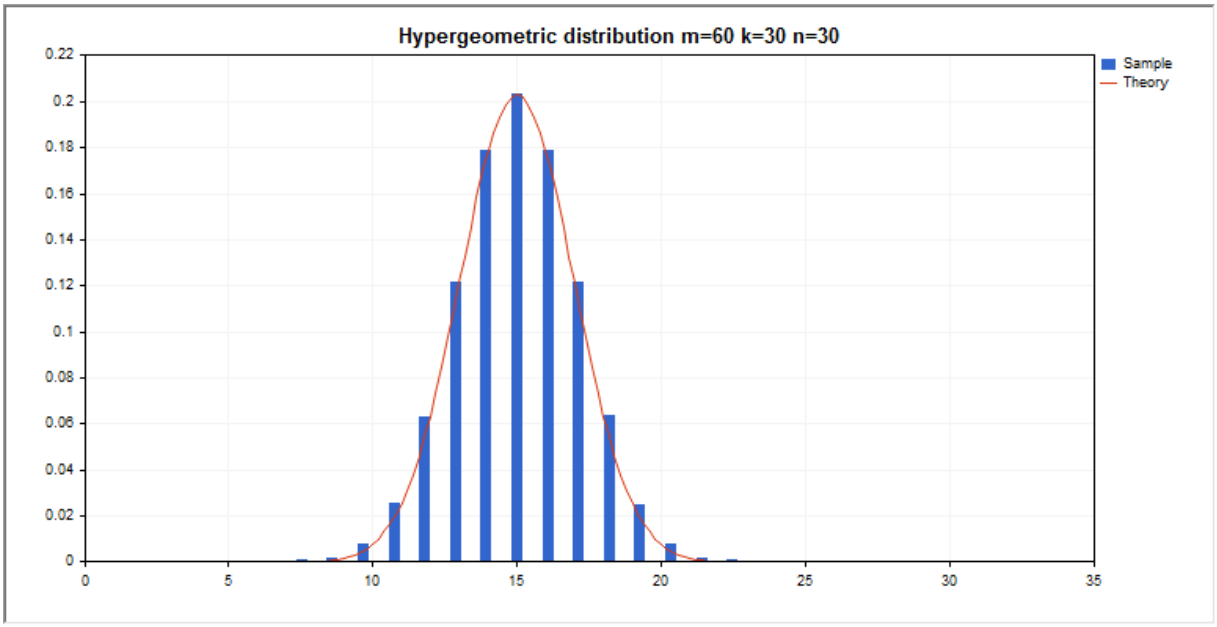
Distribuzione ipergeometrica

Questa sezione contiene funzioni per lavorare con distribuzione ipergeometrica. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge ipergeometrica. La distribuzione ipergeometrica è definita dalla seguente formula:

$$f_{Hypergeometric}(x \mid m,k,n) = \frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

dove:

- x — valore della variabile casuale (intero)
- m — numero totale di oggetti
- k — numero di oggetti con la caratteristica desiderata
- n — numero di oggetti disegna



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityHypergeometric	Calcola la funzione di densità di probabilità della distribuzione ipergeometrica
MathCumulativeDistributionHypergeometric	Calcola il valore della funzione di distribuzione di probabilità ipergeometrica
MathQuantileHypergeometric	Calcola il valore della funzione di distribuzione ipergeometrica inversa per la probabilità specificata
MathRandomHypergeometric	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di distribuzione ipergeometrica

Funzione	Descrizione
MathMomentsHypergeometric	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione ipergeometrica

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Hypergeometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double m_par=60;      // il numero totale di oggetti
input double k_par=30;      // il numero totale di oggetti con la caratteristica desi
input double n_par=30;      // il numero totale di oggetti disegnati
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // il numero di valori nell'esempio
   int ncells=15;       // il numero di intervalli nell'istogramma
   double x[];          // centro degli intervalli dell'istogramma
   double y[];          // il numero di valori dall'esempio che cade all'interno dell
   double data[];       // esempio di valori casuali
   double max,min;      // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione ipergeometrica
   MathRandomHypergeometric(m_par,k_par,n_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityHypergeometric(x2,m_par,k_par,n_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Hypergeometric distribution m=%G k=%G n=%G",m_
   graphic.BackgroundMainSize(16);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
```

```
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisi
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityHypergeometric

Calcola il valore della funzione di massa di probabilità di distribuzione ipergeometrica con i parametri M, K ed N per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // valore della variabile random (integer)
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione ipergeometrica con i parametri M, K ed N per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // valore della variabile random (integer)
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione ipergeometrica con i parametri M, K ed N per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dhyper() in R.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // array con i valori della variabile random
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di massa di probabilità di distribuzione ipergeometrica con i parametri M, K ed N per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // array con i valori della variabile random
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

m

[in] Numero totale di oggetti (integer).

k

[in] Numero di oggetti con la caratteristica desiderata (integer).

n

[in] Numero di oggetti presi (integer).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionHypergeometric

Calcola il valore della funzione di distribuzione di probabilità per legge ipergeometrica con i parametri M, K ed N per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // valore della variabile random (integer)
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge ipergeometrica con i parametri M, K ed N per una variabile casuale X. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // valore della variabile random (integer)
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge ipergeometrica con i parametri M, K ed N per un array di variabili casuali x[ ]. In caso di errore restituisce false. Analogo diphyper() in R.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // array con i valori della variabile random
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di distribuzione
   );
```

Calcola il valore della funzione di distribuzione di probabilità per legge ipergeometrica con i parametri M, K ed N per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // array con i valori della variabile random
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   double&       result[]        // array per i valori della funzione di distribuzione
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

m

[in] Numero totale di oggetti (integer).

k

[in] Numero di oggetti con la caratteristica desiderata (integer).

n

[in] Numero di oggetti presi (integer).

tail

[in] Flag di calcolo, se è true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore, se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per valori della funzione di distribuzione.

MathQuantileHypergeometric

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per legge ipergeometrica con i parametri M, K ed N. In caso di errore restituisce NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione per legge ipergeometrica con i parametri M, K ed N. In caso di errore restituisce NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per legge ipergeometrica con i parametri M, K ed N. In caso di errore restituisce false. Analogo di qhyper() in R.

```
double  MathQuantileHypergeometric(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione per legge ipergeometrica con i parametri M, K ed N. In caso di errore restituisce false.

```
bool  MathQuantileHypergeometric(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   double&       result[]        // array con i valori dei quantili
```

```
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

m

[in] Numero totale di oggetti (integer).

k

[in] Numero di oggetti con la caratteristica desiderata (integer).

n

[in] Numero di oggetti presi (integer).

tail

[in] Flag di calcolo, se tail=false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomHypergeometric

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione ipergeometrica con i parametri M, N e K. In caso di errore restituisce NaN.

```
double  MathRandomHypergeometric(
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione ipergeometrica con i parametri M, N e K. In caso di errore restituisce false. Analogo di rhyper() in R.

```
bool  MathRandomHypergeometric(
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   const int     data_count,     // ammontare dei dati richiesti
   double&       result[]        // array con i valori delle variabili pseudocasuali
   );
```

Parametri

m

[in] Numero totale di oggetti (integer).

k

[in] Numero di oggetti con la caratteristica desiderata (integer).

n

[in] Numero di oggetti presi (integer).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsHypergeometric

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione ipergeometrica con i parametri M, N e K.

```
double  MathMomentsHypergeometric(
   const double  m,              // numero totale degli oggetti (integer)
   const double  k,              // numero di oggetti con la caratteristica desiderata
   const double  n,              // numero di oggetti disegnati (integer)
   double&       mean,           // variabile per la media
   double&       variance,       // variabile per la varianza
   double&       skewness,       // variabile per l'asimmetria
   double&       kurtosis,       // variabile per la curtosi
   int&          error_code      // variabile per il codice errore
   );
```

Parametri

m

[in] Numero totale di oggetti (integer).

k

[in] Numero di oggetti con la caratteristica desiderata (integer).

n

[in] Numero di oggetti presi (integer).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

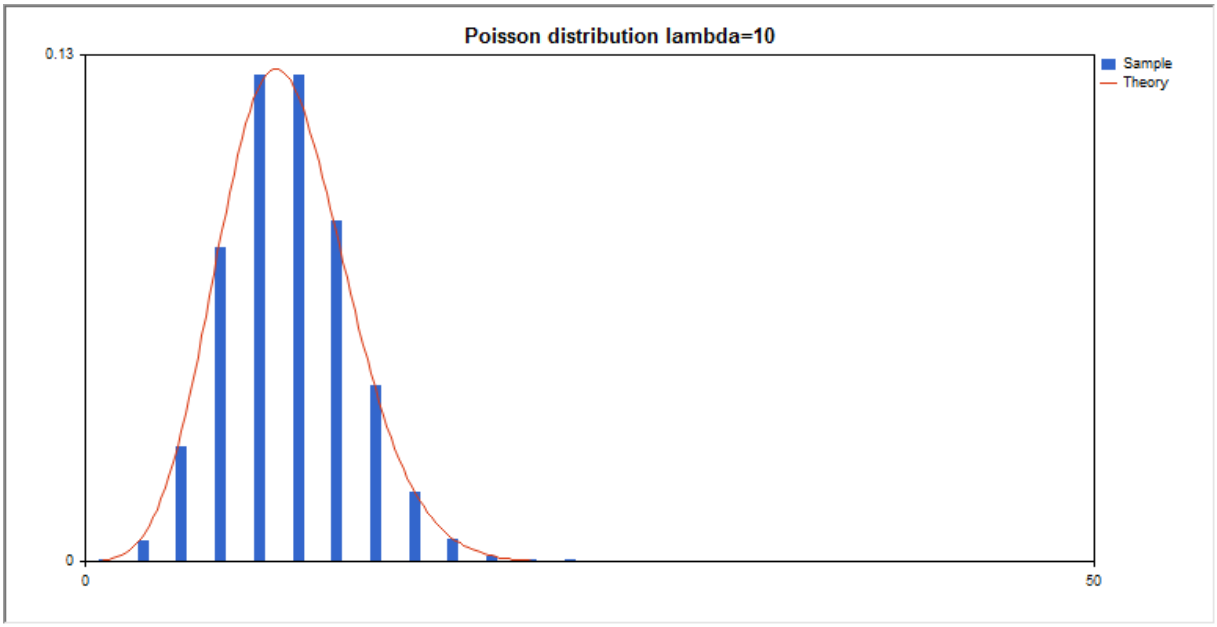
Distribuzione di Poisson

Questa sezione contiene funzioni per lavorare con la distribuzione di Poisson. Esse permettono di calcolare la densità, probabilità, quantili e generare numeri pseudo-casuali distribuiti secondo la legge di Poisson. La distribuzione di Poisson è definita dalla seguente formula:

$$f_{Poisson}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$$

dove:

- x — valore della variabile casuale
- λ — parametro della distribuzione (media)



Oltre al calcolo delle singole variabili casuali, la libreria implementa anche la capacità di lavorare con array di variabili casuali.

Funzione	Descrizione
MathProbabilityDensityPoisson	Calcola la funzione di densità di probabilità della distribuzione di Poisson
MathCumulativeDistributionPoisson	Calcola il valore della funzione di distribuzione di probabilità di Poisson
MathQuantilePoisson	Calcola il valore della funzione di distribuzione di Poisson inversa per la probabilità specificata
MathRandomPoisson	Genera una variabile/array di variabili pseudocasuali distribuite secondo la legge di Poisson
MathMomentsPoisson	Calcola i valori numerici teorici dei primi 4 momenti della distribuzione di Poisson

Esempio:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Poisson.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- parametri di input
input double lambda_par=10;      // parametro dell distribuzione (media)
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nascondere il grafico(chart) dei prezzi
   ChartSetInteger(0,CHART_SHOW,false);
//--- inizializza il generatore di numeri casuali
   MathSrand(GetTickCount());
//--- genera un esempio della variabile casuale
   long chart=0;
   string name="GraphicNormal";
   int n=100000;        // il numero di valori nel campione
   int ncells=13;       // il numero di intervalli nell'istogramma
   double x[];          // centro degli intervalli dell'istogramma
   double y[];          // il numero di valori dall'esempio che cade all'interno dell
   double data[];       // esempio di valori casuali
   double max,min;      // i valori massimo e minimo nell'esempio
//--- ottiene un campione dalla distribuzione di Poisson
   MathRandomPoisson(lambda_par,n,data);
//--- calcolare i dati per tracciare l'istogramma
   CalculateHistogramArray(data,x,y,max,min,ncells);
// --- ottenere i confini sequenza e la fase di determinazione del disegnamento della
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
/ --- ottiene i dati teoricamente calcolati in base all'intervallo di [min, max]
   double x2[];
   double y2[];
   MathSequence(0,int(MathCeil(max)),1,x2);
   MathProbabilityDensityPoisson(x2,lambda_par,false,y2);
//--- imposta la scala
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- output charts
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

```
   graphic.BackgroundMain(StringFormat("Poisson distribution lambda=%G",lambda_par));
   graphic.BackgroundMainSize(16);
//--- disabilita la scalatura automatica dell'asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,2));
   graphic.YAxis().Min(0);
//--- disegna tutte le curve
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e ora tracciare la curva teorica della densità di distribuzione
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- disegna tutte le curve
   graphic.Update();
  }
//+------------------------------------------------------------------+
//| Calcolare le frequenze per set di dati                           |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definire il centro dell'intervallo
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- riempie le frequenze di caduta all'interno dell'intervallo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//| Calcola i valori per la generazione di sequenze                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
```

```
  {
//--- calcola il range assoluto della sequenza per ottenere la precisione di normaliz
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizza i valori massimi e minimi alla precisione specificata
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- la fase di generazione di sequenza viene inoltre impostata in base alla precisio
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityPoisson

Calcola il valore della funzione di probabilità di massa della distribuzione di Poisson di parametro lambda per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // valore della variabile random (integer)
   const double  lambda,         // parametro della distribuzione (media)
   const bool    log_mode,       // calcola il logaritmo del valore, se log_mode=true,
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di probabilità di massa della distribuzione di Poisson di parametro lambda per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // valore della variabile random (integer)
   const double  lambda,         // parametro della distribuzione (media)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di probabilità di massa della distribuzione di Poisson di parametro lambda per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di dpois() in R.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // array con i valori della variabile random
   const double  lambda,         // parametro della distribuzione (media)
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Calcola il valore della funzione di probabilità di massa della distribuzione di Poisson col parametro lambda per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // array con i valori della variabile random
   const double  lambda,         // parametro della distribuzione (media)
   double&       result[]        // array per i valori della funzione di densita di pr
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

lambda

[in] Parametro della distribuzione (media).

log_mode

[in] Flag per calcolare il logaritmo del valore. Se log_mode=true, allora viene restituito il logaritmo naturale della densità di probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per i valori della funzione di densità di probabilità.

MathCumulativeDistributionPoisson

Calcola il valore della funzione di distribuzione di Poisson con parametro lambda per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // valore della variabile random (integer)
   const double  lambda,         // parametro della distribuzione (media)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata la
   const bool    log_mode,       // flag per calcolare il logaritmo del valore, se log_
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di Poisson con parametro lambda per una variabile casuale x. In caso di errore restituisce NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // valore della variabile random (integer)
   const double  lambda,         // parametro della distribuzione (media)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Calcola il valore della funzione di distribuzione di Poisson con il parametro lambda per una serie di variabili casuali x[ ]. In caso di errore restituisce false. Analogo di ppois() in R.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // array con i valori della variabile random
   const double  lambda,         // parametro della distribuzione (media)
   const bool    tail,           // flag di calcolo, se true, allora viene calcolata l
   const bool    log_mode,       // flag di calcolo del logaritmo del valore, se log_m
   double&       result[]        // array per i valori della funzione di distribuzione
   );
```

Calcola il valore della funzione di distribuzione di Poisson con parametro lambda per una serie di variabili casuali x[ ]. In caso di errore restituisce false.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // array con i valori della variabile random
   const double  lambda,         // parametro della distribuzione (media)
   double&       result[]        // array per i valori della funzione di distribuzione
   );
```

Parametri

x

[in] Valore della variabile random.

x[]

[in] Array con i valori della variabile random.

lambda

[in] Parametro della distribuzione (media).

tail

[in] Flag di calcolo, se è true, allora viene calcolata la probabilità di variabile casuale non superiore x.

log_mode

[in] Flag per calcolare il logaritmo del valore, se log_mode=true, allora viene calcolato il logaritmo naturale della probabilità.

error_code

[out] variabile per memorizzare il codice di errore.

result[]

[out] Array per valori della funzione di distribuzione.

MathQuantilePoisson

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione di Poisson col parametro lambda. In caso di errore restituisce NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  lambda,         // parametro della distribuzione (media)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per la specificata *probabilità*, la funzione calcola il valore inverso della funzione di distribuzione di Poisson col parametro lambda. In caso di errore restituisce NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // valore probabilità dell'occorrenza della variabile
   const double  lambda,         // parametro della distribuzione (media)
   int&          error_code      // variabile per memorizzare il codice errore
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione di Poisson con il parametro lambda. In caso di errore restituisce false. Analogo di qpois() in R.

```
double  MathQuantilePoisson(
   const double& probability[],  // array con i valori della proabilità della variabil
   const double  lambda,         // parametro della distribuzione (media)
   const bool    tail,           // flag di calcolo, se false, allora il calcolo viene
   const bool    log_mode,       // flag di calcolo, se log_mode=true, il calcolo vier
   double&       result[]        // array con i valori dei quantili
   );
```

Per lo specificato array *probability[]* di valori di probabilità, la funzione calcola il valore inverso della funzione di distribuzione di Poisson con il parametro lambda. In caso di errore restituisce false.

```
bool  MathQuantilePoisson(
   const double& probability[],  // array i valori della probabilità della variabile r
   const double  lambda,         // parametro della distribuzione (media)
   double&       result[]        // array con i valori dei quantili
   );
```

Parametri

probability

[in] Valore Probabilità della variabile casuale.

probability[]

[an] Array con i valori di probabilità di una variabile casuale.

lambda

[in] Parametro della distribuzione (media).

tail

[in] Flag di calcolo, se tail=false, allora il calcolo viene eseguito per 1.0-probabilità.

log_mode

[in] Flag di calcolo, se log_mode=true, il calcolo viene eseguito per Exp(probabilità).

error_code

[out] Variabile per ottenere il codice di errore.

result[]

[out] Array con valori di quantili.

MathRandomPoisson

Genera una variabile pseudocasuale distribuita secondo la legge di distribuzione di Poisson con il parametro lambda. In caso di errore restituisce NaN.

```
double  MathRandomPoisson(
   const double  lambda,          // parametro della distribuzione (media)
   int&          error_code       // variabile per memorizzare il codice errore
   );
```

Genera variabili pseudocasuali distribuite secondo la legge di distribuzione di Poisson con il parametro lambda. In caso di errore restituisce false. Analogo di rpois() in R.

```
bool  MathRandomPoisson(
   const double  lambda,          // parametro della distribuzione (media)
   const int     data_count,      // ammontare dei dati richiesti
   double&       result[]         // array con i valori delle variabili pseudocasuali
   );
```

Parametri

lambda

[in] Parametro della distribuzione (media).

error_code

[out] variabile per memorizzare il codice di errore.

data_count

[out] Ammontare dei dati richiesti.

result[]

[out] Array per ottenere i valori delle variabili pseudocasuali.

MathMomentsPoisson

Calcola i valori numerici teorici dei primi 4 momenti della distribuzione di Poisson con parametro lambda.

```
double  MathMomentsPoisson(
   const double  lambda,          // parametro della distribuzione (media)
   double&       mean,            // variabile per la media
   double&       variance,        // variabile per la varianza
   double&       skewness,        // variabile per l'asimmetria
   double&       kurtosis,        // variabile per la curtosi
   int&          error_code       // variabile per il codice errore
   );
```

Parametri

lambda

[in] Parametro della distribuzione (media).

mean

[out] Variabile per ottenere il valore medio.

variance

[out] variabile per ottenere la varianza.

skewness

[out] variabile per ottenere l' asimmetria.

kurtosis

[out] Variabile per ottenere la curtosi.

error_code

[out] Variabile per ottenere il codice di errore.

Valore di ritorno

Restituisce true se il calcolo dei momenti ha avuto successo, altrimenti false.

Subfunzioni

Gruppo di funzioni che eseguono operazioni matematiche di base: calcolo della funzione gamma, funzione beta, fattoriali, esponenziali, logaritmi con basi diverse, radice quadrata, ecc

Esse forniscono la capacità di elaborare sia i singoli valori numerici (real ed integer) ed array di tali valori (con output dei risultati ad un array separato o originale).

Funzione	Descrizione
MathRandomNonZero	Restituisce un numero casuale con un il floating point nell'intervallo da 0.0 a 1.0.
MathMoments	Calcola i primi 4 momenti di elementi dell'array: media, varianza, asimmetria, curtosi.
MathPowInt	Eleva un numero alla potenza intera specificata.
MathFactorial	Calcola il fattoriale del numero intero specificato.
MathTrunc	Calcola la parte intera del numero o elementi array specificati.
MathRound	Arrotonda un numero o una serie di numeri per il numero specificato di cifre decimali.
MathArctan2	Calcola l'angolo, la tangente che è uguale al rapporto tra i due numeri specificati nell'intervallo [-pi, pi].
MathGamma	Calcola il valore della funzione gamma.
MathGammaLog	Calcola il logaritmo della funzione gamma.
MathBeta	Calcola il valore della funzione beta.
MathBetaLog	Calcola il logaritmo della funzione beta.
MathBetaIncomplete	Calcola il valore della funzione beta incompleta.
MathGammaIncomplete	Calcola il valore della funzione gamma incompleta.
MathBinomialCoefficient	Calcola il coefficiente binomiale.
MathBinomialCoefficientLog	Calcola il logaritmo del coefficiente binomiale.
MathHypergeometric2F2	Calcola il valore della funzione ipergeometrica.
MathSequence	Genera una sequenza basata su valori: il primo elemento, l'ultimo elemento, il passo della sequenza.

Funzione	Descrizione
MathSequenceByCount	Genera una sequenza basata su valori: il primo elemento, l'ultimo elemento, il numero di elementi nella sequenza.
MathReplicate	Genera una sequenza di valori.
MathReverse	Genera un'array di valori con ordine inverso di elementi.
MathIdentical	Confronta due array di valori e restituisce true se tutti gli elementi corrispondono.
MathUnique	Genera un array con solo valori univoci.
MathQuickSortAscending	Funzione per l'ordinamento in ordine crescente.
MathQuickSortDescending	Funzione per l'ordinamento in ordine decrescente.
MathQuickSort	Funzione per l'ordinamento.
MathOrder	Genera un array con permutazione secondo l' ordine degli elementi dell'array dopo l'ordinamento.
MathBitwiseNot	Calcola il risultato di un'operazione NOT di bit per elementi dell'array.
MathBitwiseAnd	Calcola il risultato di un'operazione AND di bit per elementi dell'array.
MathBitwiseOr	Calcola il risultato di un'operazione OR di bit per elementi dell'array.
MathBitwiseXor	Calcola il risultato di un'operazione XOR di bit per elementi dell'array.
MathBitwiseShiftL	Calcola il risultato dell'operazione bit SHL per elementi dell'array.
MathBitwiseShiftR	Calcola il risultato dell'operazione bit SHR per elementi dell'array.
MathCumulativeSum	Genera un array con somme cumulative.
MathCumulativeProduct	Genera un array con i prodotti cumulativi.
MathCumulativeMin	Genera un array con i minimi cumulativi.
MathCumulativeMax	Genera un array con i massimi cumulativi.
MathSin	Calcola i valori della funzione sin(x) per elementi dell'array.

Funzione	Descrizione
MathCos	Calcola i valori della funzione cos(x) per elementi dell'array.
MathTan	Calcola i valori della funzione tan(x) per elementi dell'array.
MathArcsin	Calcola i valori della funzione arcsin(x) per elementi dell'array.
MathArccos	Calcola i valori della funzione arccos(x) per elementi dell'array.
MathArctan	Calcola i valori della funzione arctan(x) per elementi dell'array.
MathSinPi	Calcola i valori della funzione sin(pi*x) per elementi dell'array.
MathCosPi	Calcola i valori dei cos (pi*x) per elementi dell'array.
MathTanPi	Calcola i valori della funzione tan(pi*x) per elementi dell'array.
MathAbs	Calcola i valori assoluti degli elementi dell'array.
MathCeil	Restituisce i numeri interi più grandi più vicini per gli elementi dell'array.
MathFloor	Restituisce i numeri interi più piccoli più vicini per gli elementi dell'array.
MathSqrt	Calcola la radice quadrata di elementi dell'array.
MathExp	Calcola i valori della funzione exp(x) per elementi dell'array.
MathPow	Calcola i valori della funzione pow(x, potenza) per elementi dell'array.
MathLog	Calcola i valori della funzione log(x) per elementi dell'array.
MathLog2	Calcola il logaritmo in base 2 per gli elementi dell'array.
MathLog10	Calcola il logaritmo in base 10 per gli elementi dell'array.
MathDifference	Genera un array con differenze elemento di y[i]=x[i+lag]-x[i].
MathSample	Genera un campione casuale dagli elementi dell'array.

Funzione	Descrizione
MathTukeySummary	Calcola la sintesi a cinque-numeri di Tukey per gli elementi dell'array.
MathRange	Calcola i valori minimi e massimi di elementi array.
MathMin	Restituisce il valore minimo di tutti gli elementi dell'array.
MathMax	Restituisce il valore massimo di tutti gli elementi dell'array.
MathSum	Restituisce la somma degli elementi dell'array.
MathProduct	Restituisce il prodotto di elementi array.
MathStandardDeviation	Calcola la deviazione standard di elementi array.
MathAverageDeviation	Calcola la deviazione media assoluta di elementi array.
MathMedian	Calcola il valore mediano di elementi dell'array.
MathMean	Calcola i valori medi di elementi dell'array.
MathVariance	Calcola la varianza degli elementi dell'array.
MathSkewness	Calcola l'asimmetria degli elementi dell'array.
MathKurtosis	Calcola la curtosi degli elementi dell'array.
MathLog1p	Calcola i valori del log(1+x) per elementi dell'array.
MathExpm1	Calcola i valori della funzione exp(x)-1 di elementi dell'array.
MathSinh	Calcola i valori della funzione sinh(x) per elementi dell'array.
MathCosh	Calcola i valori della funzione cosh(x) per elementi dell'array.
MathTanh	Calcola i valori della funzione tanh(x) per elementi dell'array.
MathArcsinh	Calcola i valori della funzione arcsinh(x) per elementi dell'array.
MathArccosh	Calcola i valori della funzione arccosh(x) per elementi dell'array.
MathArctanh	Calcola i valori della funzione arctanh(x) per elementi dell'array.

Funzione	Descrizione
MathSignif	Arrotonda un valore per il numero specificato di cifre nella mantissa.
MathRank	Calcola le fila di elementi dell'array.
MathCorrelationPearson	Calcola il coefficiente di correlazione di Pearson.
MathCorrelationSpearman	Calcola il coefficiente di correlazione di Spearman.
MathCorrelationKendall	Calcola il coefficiente di correlazione di Kendall.
MathQuantile	Calcola quantili campione corrispondenti alle probabilità specificata.
MathProbabilityDensityEmpirical	Calcola la funzione di densità di probabilità empirica per valori casuali.
MathCumulativeDistributionEmpirical	Calcola la funzione di distribuzione cumulativa empirica per valori casuali.

MathRandomNonZero

Restituisce un numero casuale con un il floating point nell'intervallo da 0.0 a 1.0.

```
double  MathRandomNonZero()
```

Valore di ritorno

Numero casuale con virgola mobile nell'intervallo da 0.0 a 1.0.

MathMoments

Calcola i primi 4 momenti di elementi dell'array: media, varianza, asimmetria, curtosi.

```
bool  MathMoments(
   const double&  array[],            // array dei valori
   double&        mean,               // variabile per la media
   double&        variance,           // variabile per la varianza
   double&        skewness,           // variabile per l'asimmetria
   double&        kurtosis,           // variabile per la curtosi
   const int      start=0,            // indice iniziale
   const int      count=WHOLE_ARRAY   // il numero di elementi
  );
```

Parametri

array[]

[in] Array deivalori.

mean

[out] Variabile per la media (1mo momento).

variance

[out] Variabile per la varianza (2ndo momento).

skewness

[out] variabile per l'asimmetria (3zo momento).

kurtosis

[out] variabile per la curtosi (4to momento).

start=0

[in] indice iniziale per il calcolo.

count=WHOLE_ARRAY

[in] Il numero di elementi per il calcolo.

Valore di ritorno

Restituisce true se i momenti sono stati calcolati con successo, altrimenti false.

MathPowInt

Eleva un numero alla potenza intera specificata.

```
double  MathPowInt(
   const double  x,      // valore del numero
   const int     power   // potenza a cui elevare
  );
```

Parametri

x

[in] Numero in virgola mobile a doppia precisione da elevare alla potenza.

power

[in] Intero che specifica la potenza.

Valore di ritorno

Il numero *X* elevato alla potenza specificata.

MathFactorial

Calcola il fattoriale del numero intero specificato.

```
double  MathFactorial(
   const int  n   // valore del numero
  );
```

Parametri

n

[in] Numero integer, il fattoriale che deve essere calcolato.

Valore di ritorno

Il fattoriale del numero.

MathTrunc

Calcola la parte intera del numero o elementi array specificati.

Versione per lavorare con un numero a virgola mobile a doppia precisione:

```
double  MathTrunc(
   const double   x          // valore del numero
  );
```

Valore di ritorno

La parte intera(integer) del numero specificato.

Versione per lavorare con un array di numeri in virgola mobile a doppia precisione. I risultati vengono dati in output in un array:

```
bool  MathTrunc(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Versione per lavorare con un array di numeri in virgola mobile a doppia precisione. I risultati sono dati in output all'array originale:

```
bool  MathTrunc(
   double&        array[]    // array di valori
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Parametri

x

[in] Numero in virgola mobile a doppia precisione, la parte intera che si intende ottenere.

array[]

[in] Array di numeri in virgola mobile a doppia precisione, le parti integer di numeri che si intendono ottenere.

array[]

[out] Array di valori di uscita.

result[]

[out] Array di valori di uscita.

MathRound

Arrotonda un numero in virgola mobile a doppia precisione o un array di tali numeri per il numero specificato di cifre decimali.

Versione per arrotondamento a doppia precisione numero a virgola mobile al numero specificato di cifre decimali:

```
double  MathRound(
   const double   x,          // valore del numero
   const int      digits      // il numero di posizioni decimali
  );
```

Valore di ritorno

Un numero più vicino al parametro x, con il numero di posizioni decimali uguale alla cifra.

Versione per arrotondare un array di numeri in virgola mobile a doppia precisione al numero specificato di cifre decimali. I risultati vengono dati in output in un array.

```
bool  MathRound(
   const double&  array[],    // array di valori
   int            digits,     // il numero di posizioni decimali
   double&        result[]    // array dei risultati
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Versione per arrotondare un array di numeri in virgola mobile a doppia precisione al numero specificato di cifre decimali. I risultati sono dati in output all' array originale.

```
bool  MathRound(
   double&        array[],    // array di valori
   int            digits      // il numero di posizioni decimali
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Parametri

x

[in] Numero in virgola mobile a doppia precisione da arrotondare.

digits

[in] Il numero di cifre decimali in valore restituito.

array[]

[in] Array di numeri in virgola mobile a doppia precisione da arrotondare.

array[]

[out] Array di valori output.

result[]

[out] Array di valori output.

MathArctan2

Restituisce l'arcotangente del quoziente di due argomenti (x, y).

Versione per lavorare con il rapporto dei due numeri specificati (x, y):

```
double  MathArctan2(
   const double       y,          // Coordinate Y
   const double       x           // Coordinate X
  );
```

Valore di ritorno

Angolo θ, misurato in radianti, cosicché -π≤θ≤π e tan (θ) = y oppure x, dove (x, y) è un punto in un sistema di coordinate Cartesiane.

Versione per lavorare con il rapporto delle coppie di elementi dagli array x ed y:

```
bool  MathArctan2(
   const double&      x[],        // array di valori x
   const double&      y[],        // array di valori y
   double&            result[]    // array di risultati
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Parametri

y

[in] Le coordinate Y del punto.

x

[in] Le coordinate X del punto.

x[]

[in] Array di coordinate X dei punti.

y[]

[in] Array di coordinate Y dei punti.

result[]

[out] Array per dare in output i risultati

Note

Si prega di notare quanto segue.

- Per (x, y) nel quadrante 1, il valore restituito sarà: 0 < θ < π/2.
- Per (x, y) nel quadrante 2, il valore restituito sarà: π/2 <θ≤π.
- Per (x, y) nel quadrante 3, il valore restituito sarà: -π <θ <-π/2.
- Per (x, y) nel quadrante 4, il valore restituito sarà: -π/2 <θ <0.

Il valore di ritorno per i punti fuori questi quadranti è indicato di seguito.

- Se y è 0 e x non è negativo, allora θ = 0.
- Se y è 0 e x è negativo, allora θ = π.
- Se y è un numero positivo, e x è 0, allora θ = π/2.
- Se y è negativo e x è 0, allora θ = -π/2.
- Se y è 0 e x è 0, allora θ = -π/2.

Se il valore del parametro X o Y è NaN, o se i valori dei parametri x ed y sono uguali al valore PositiveInfinity o NegativeInfinity, il metodo restituisce il valore NaN.

MathGamma

Calcola il valore della funzione gamma per l'argomento real *X*.

```
double  MathGamma(
   const double  x       // argomento della funzione
  );
```

Parametri

x

[in] L'argomento real della funzione.

Valore di ritorno

Il valore della funzione gamma.

MathGammaLog

Calcola il logaritmo della funzione gamma per l'argomento reale x.

```
double  MathGammaLog(
   const double  x      // argomento della funzione
  );
```

Parametri

x

[in] L'argomento real della funzione.

Valore di ritorno

Logaritmo della funzione.

MathBeta

Calcola il valore della funzione beta per gli argomenti reali a e b.

```
double  MathBeta(
   const double  a,      // il primo argomento della funzione
   const double  b       // il secondo argomento della funzione
  );
```

Parametri

a

[in] L' argomento a della funzione.

b

[in] L' argomento b della funzione.

Valore di ritorno

Valore della funzione.

MathBetaLog

Calcola il logaritmo della funzione beta per gli argomenti reali a e b.

```
double  MathBetaLog(
   const double  a,      // il primo argomento della funzione
   const double  b       // il secondo argomento della funzione
  );
```

Parametri

a

[in] L' argomento a della funzione.

b

[in] L' argomento b della funzione.

Valore di ritorno

Logaritmo della funzione.

MathBetaIncomplete

Calcola il valore della funzione beta incompleta.

```
double  MathBetaIncomplete(
   const double  x,      // argomento della funzione
   const double  p,      // il primo parametro della funzione
   const double  q       // il secondo parametro della funzione
  );
```

Parametri

x

[in] L'argomento della funzione.

p

[in] Il primo parametro della funzione beta, deve essere > 0.0.

q

[in] Il secondo parametro della funzione beta, deve essere > 0.0.

Valore di ritorno

Valore della funzione.

MathGammaIncomplete

Calcola il valore della funzione gamma incompleta.

```
double  MathGammaIncomplete(
   double  x,          // argomento della funzione
   double  alpha       // parametro della funzione
  );
```

Parametri

x

[in] L'argomento della funzione.

alpha

[in] Il parametro della funzione gamma incompleta.

Valore di ritorno

Valore della funzione.

MathBinomialCoefficient

Calcola il coefficiente binomiale: C(n,k)=n!/(k!*(n-k)!)..

```
long  MathBinomialCoefficient(
   const int  n,       // il numero totale degli elementi
   const int  k        // il numero di elementi in combinazione
  );
```

Parametri

n

[in] Il numero di elementi.

k

[in] Il numero di elementi per ciascuna combinazione.

Valore di ritorno

Il numero di combinazioni di N scelto K.

MathBinomialCoefficientLog

Calcola il logaritmo del coefficiente binomiale: Log(C(n,k))=Log(n!/(k!*(n-k)!))

Versione per argomenti integer:

```
double  MathBinomialCoefficientLog(
   const int      n,       // il numero totale degli elementi
   const int      k        // il numero di elementi in combinazione
  );
```

Versione per argomenti reali:

```
double  MathBinomialCoefficientLog(
   const double   n,       // il numero totale di elementi
   const double   k        // il numero totale di elementi in combinazione
  );
```

Parametri

n

[in] Il numero di elementi.

k

[in] Il numero di elementi per ciascuna combinazione.

Valore di ritorno

Il logaritmo di C(n,k).

MathHypergeometric2F2

Calcola il valore della funzione Hypergeometric_2F2 (a, b, c, d, z) utilizzando il metodo di Taylor.

```
double  MathHypergeometric2F2(
   const double  a,      // il primo parametro della funzione
   const double  b,      // il secondo parametro della funzione
   const double  c,      // il terzo parametro della funzione
   const double  d,      // il quarto parametro della funzione
   const double  z       // il quarto parametro della funzione
  );
```

Parametri

a

[in] Il primo parametro della funzione.

b

[in] Il secondo parametro della funzione.

c

[in] Il terzoparametro della funzione.

d

[in] Il quarto parametro della funzione.

z

[in] Il quinto parametro della funzione.

Valore di ritorno

Valore della funzione.

MathSequence

Genera una sequenza di valori in base ai seguenti valori: il primo elemento, l'ultimo elemento, il passo della sequenza.

Versione per lavorare con valori reali:

```
bool  MathSequence(
   const double  from,        // valore iniziale
   const double  to,          // valore finale
   const double  step,        // step
   double&       result[]     // array dei risultati
  );
```

Versione per lavorare con valori interi:

```
bool  MathSequence(
   const int     from,        // valore iniziale
   const int     to,          // valore finale
   const int     step,        // step
   int&          result[]     // array dei risultati
  );
```

Parametri

from

[in] Il primo valore della sequenza

to

[in] L'ultimo valore della sequenza

step

[in] Il passo della sequenza.

result[]

[out] Array dove dare in output la sequenza.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathSequenceByCount

Genera una sequenza di valori in base ai seguenti valori: il primo elemento, l'ultimo elemento, il numero di elementi nella sequenza.

Versione per lavorare con valori reali:

```
bool  MathSequenceByCount(
   const double  from,        // valore iniziale
   const double  to,          // valore finale
   const int     count,       // conteggio
   double&       result[]     // array dei risultati
  );
```

Versione per lavorare con valori interi:

```
bool  MathSequenceByCount(
   const int     from,        // valore iniziale
   const int     to,          // valore finale
   const int     count,       // conteggio
   int&          result[]     // array dei risultati
  );
```

Parametri

from

[in] Il primo valore della sequenza.

to

[in] L'ultimo valore della sequenza.

count

[in] Il numero di elementi nella sequenza.

result[]

[out] Array dove dare in output la sequenza.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathReplicate

Genera una sequenza di valori.

Versione per lavorare con valori reali:

```
bool  MathReplicate(
   const double&  array[],    // array di valori
   const int      count,      // numero di ripetizioni
   double&        result[]    // array dei risultati
  );
```

Versione per lavorare con valori interi:

```
bool  MathReplicate(
   const int&     array[],    // array di valori
   const int      count,      // numero di ripetizioni
   int&           result[]    // array dei risultati
  );
```

Parametri

array[]

[in] Array per generare una sequenza.

count

[in] Il numero delle ripetizioni dell'array nella sequenza.

result[]

[out] Array dove dare in output la sequenza.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathReverse

Genera un'array di valori con ordine inverso di elementi.

Versione per lavorare con valori reali e con l'output dei risultati in un nuovo array:

```
bool  MathReverse(
   const double&  array[],      // array dei valori
   double&        result[]      // array dei risultati
  );
```

Versione per lavorare con valori interi e con l'output dei risultati in un nuovo array:

```
bool  MathReverse(
   const int&     array[],      // array dei valori
   int&           result[]      // array dei risultati
  );
```

Versione per lavorare con valori real e con l'output dei risultati all'array originale.

```
bool  MathReverse(
   double&        array[]       // array di valori
  );
```

Versione per lavorare con valori interi e con l'output dei risultati all'array originale:

```
bool  MathReverse(
   int&           array[]       // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

array[]

[out] Array di output con l'ordine inverso di valori.

result[]

[out] Array di output con l'ordine inverso di valori.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathIdentical

Confronta due array di valori e restituisce true se tutti gli elementi corrispondono.

Versione per lavorare con array di valori reali:

```
bool  MathIdentical(
   const double&  array1[],      // il primo array dei valori
   const double&  array2[]       // il secondo array dei valori
  );
```

Versione per lavorare con array di valori interi:

```
bool  MathIdentical(
   const int&     array1[],      // il primo array dei valori
   const int&     array2[]       // il secondo array dei valori
  );
```

Parametri

array1[]

[in] Il primo array da confrontare.

array2[]

[in] Il secondo array da confrontare.

Valore di ritorno

Restituisce true se gli array sono uguali, altrimenti false.

MathUnique

Genera un array con solo valori univoci.

Versione per lavorare con valori reali:

```
bool  MathUnique(
   const double&   array[],   // array di valori
   double&         result[]   // array di risultati
  );
```

Versione per lavorare con valori interi:

```
bool  MathUnique(
   const int&     array[],      // array dei valori
   int&           result[]      // array dei risultati
  );
```

Parametri

array[]

[in] L'array di origine.

result[]

[out] Array per dare in output i valori univoci.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathQuickSortAscending

La funzione per l'ordinamento simultaneo crescente dell'array[] ed indici[] array utilizzando l'algoritmo QuickSort.

```
void  MathQuickSortAscending(
   double&  array[],     // array di valori
   int&     indices[],   // array di indici
   int      first,       // valore iniziale
   int      last         // valore finale
  );
```

Parametri

array[]

[in][out] Array da ordinare.

indices[]

[in][out] Array per memorizzare gli indici dell'array originale.

first

[in] indice dell'elemento da cui iniziare l'ordinamento.

last

[in] indice dell'elemento dove fermare l'ordinamento.

MathQuickSortDescending

La funzione per l'ordinamento simultaneo decrescente dell'array[] ed indici[] array utilizzando l'algoritmo QuickSort.

```
void  MathQuickSortDescending(
   double&  array[],     // array di valori
   int&     indices[],   // array di indici
   int      first,       // valore iniziale
   int      last         // valore finale
  );
```

Parametri

array[]

[in][out] Array da ordinare.

indices[]

[in][out] Array per memorizzare gli indici dell'array originale.

first

[in] indice dell'elemento da cui iniziare l'ordinamento.

last

[in] indice dell'elemento dove fermare l'ordinamento.

MathQuickSort

La funzione per l'ordinamento simultaneo dell'array[] ed indici[] array utilizzando l'algoritmo QuickSort.

```
void  MathQuickSort(
   double&  array[],      // array di valori
   int&     indices[],    // array di indici
   int      first,        // valore iniziale
   int      last,         // valore finale
   int      mode          // direzione
  );
```

Parametri

array[]

[in][out] Array da ordinare.

indices[]

[in][out] Array per memorizzare gli indici dell'array originale.

first

[in] indice dell'elemento da cui iniziare l'ordinamento.

last

[in] indice dell'elemento dove fermare l'ordinamento.

mode

[in] Direzione dell'ordinamento (>0 crescente, altrimenti, decrescente).

MathOrder

Genera un array integer con permutazione secondo l'ordine degli elementi dell'array dopo l'ordinamento.

Versione per lavorare con una serie di valori reali:

```
bool  MathOrder(
   const double&  array[],   // array di valori
   int&           result[]   // array dei risultati
  );
```

Versione per lavorare con un array di valori interi:

```
bool  MathOrder(
   const int&     array[],   // array di valori
   int&           result[]   // array dei risultati
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array per l'output degli indici ordinati.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathBitwiseNot

Calcola il risultato di un'operazione NOT di bit per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathBitwiseNot(
   const int&  array[],   // array di valori
   int&        result[]   // array dei risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathBitwiseNot(
   int&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

array[]

[out] Array di valori di uscita.

result[]

[out] Array di valori di output.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathBitwiseAnd

Calcola il risultato di un'operazione AND di bit per gli array specificati.

```
bool  MathBitwiseAnd(
   const int&  array1[],  // il primo array di valori
   const int&  array2[],  // il secondo array di valori
   int&        result[]   // array dei risultati
  );
```

Parametri

array1[]

[in] Il primo array di valori.

array2[]

[in] Il secondo array di valori.

result[]

[out] Array per dare in output i risultati.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathBitwiseOr

Calcola il risultato di un'operazione OR di bit per gli array specificati.

```
bool  MathBitwiseOr(
   const int&  array1[],  // il primo array di valori
   const int&  array2[],  // il secondo array di valori
   int&        result[]   // array dei risultati
  );
```

Parametri

array1[]

[in] Il primo array di valori.

array2[]

[in] Il secondo array di valori.

result[]

[out] Array per dare in output i risultati.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathBitwiseXor

Calcola il risultato di bit XOR per gli array specificati.

```
bool  MathBitwiseXor(
   const int&  array1[],  // il primo array di valori
   const int&  array2[],  // il secondo array dei valori
   int&        result[]   // array dei risultati
  );
```

Parametri

array1[]

[in] Il primo array di valori.

array2[]

[in] Il secondo array di valori.

result[]

[out] Array per dare in output i risultati.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathBitwiseShiftL

Calcola il risultato dell'operazione di bit SHL (spostamento bit a sinistra) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathBitwiseShiftL(
   const int&  array[],    // array di valori
   const int   n,          // valore di slittamento
   int&        result[]    // array dei risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathBitwiseShiftL(
   int&        array[],    // array di valori
   const int   n           // valore di slittamento
  );
```

Parametri

array[]

[in] Array di valori.

n

[in] Il numero di bit da spostare.

array[]

[out] Array di valori di output.

result[]

[out] Array di valori di output.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathBitwiseShiftR

Calcola il risultato dell'operazione di bit SHR (spostamento bit a destra) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathBitwiseShiftR(
   const int&  array[],    // array di valori
   const int   n,          // valore di slittamento
   int&        result[]    // array dei risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathBitwiseShiftR(
   int&        array[],    // array di valori
   const int   n           // valore di slittamento
  );
```

Parametri

array[]

[in] Array di valori.

n

[in] Il numero di bit da spostare.

array[]

[out] Array di valori di output.

result[]

[out] Array di valori di output.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCumulativeSum

Genera un array con somme cumulative.

Versione con output dei risultati in un nuovo array:

```
bool  MathCumulativeSum(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCumulativeSum(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

array[]

[out] Array di valori di uscita.

result[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCumulativeProduct

Genera un array con i prodotti cumulativi.

Versione con output dei risultati in un nuovo array:

```
bool  MathCumulativeProduct(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCumulativeProduct(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCumulativeMin

Genera un array con i minimi cumulativi.

Versione con output dei risultati in un nuovo array:

```
bool  MathCumulativeMin(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCumulativeMin(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCumulativeMax

Genera un array con i massimi cumulativi.

Versione con output dei risultati in un nuovo array:

```
bool  MathCumulativeMax(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCumulativeMax(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathSin

Calcola i valori della funzione sin(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathSin(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathSin(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCos

Calcola i valori della funzione cos(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathCos(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCos(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathTan

Calcola i valori della funzione tan(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathTan(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathTan(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathArcsin

Calcola i valori della funzione arcsin(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathArcsin(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathArcsin(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathArccos

Calcola i valori della funzione arccos(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathArccos(
   const double&   array[],    // array di valori
   double&         result[]    // array dei risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathArccos(
   double&         array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathArctan

Calcola i valori della funzione arctan(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathArctan(
   const double&   array[],   // array di valori
   double&         result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathArctan(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathSinPi

Calcola i valori della funzione sin(pi*x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathSinPi(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathSinPi(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCosPi

Calcola i valori dei cos (pi*x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathCosPi(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCosPi(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathTanPi

Calcola i valori della funzione tan(pi*x) per elementi dell'array.

Versione con l'output del risultato di un nuovo array:

```
bool  MathTanPi(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con l'output del risultato all'array originale:

```
bool  MathTanPi(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathAbs

Calcola i valori assoluti degli elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathAbs(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathAbs(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCeil

Restituisce i numeri interi più grandi più vicini per gli elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathCeil(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCeil(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathFloor

Restituisce i numeri interi più piccoli più vicini per gli elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathFloor(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathFloor(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathSqrt

Calcola la radice quadrata di elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathSqrt(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathSqrt(
   double&         array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathExp

Calcola i valori della funzione exp(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathExp(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathExp(
   double&       array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathPow

Calcola i valori della funzione pow(x, potenza) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathPow(
   const double&  array[],    // array di valori
   const double   power,      // potenza
   double&        result[]    // array dei risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathPow(
   double&        array[],    // array dei valori
   const double   power       // potenza
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathLog

Calcola i valori della funzione log(x) per elementi dell'array.

Versione per il calcolo del logaritmo naturale con l'output dei risultati di un nuovo array.

```
bool  MathLog(
   const double&  array[],      // array di valori
   double&        result[]      // array di risultati
  );
```

Versione per il calcolo del logaritmo naturale con l'output dei risultati alla all'array originale.

```
bool  MathLog(
   double&        array[]       // array di valori
  );
```

Versione per calcolare il logaritmo in base specificata con l'output dei risultati in un nuovo array.

```
bool  MathLog(
   const double&   array[],     // array di valori
   const double    base,        // base del logaritmo
   double&         result[]     // array dei risultati
  );
```

Versione per calcolare il logaritmo in base specificata con l'output dei risultati nell'array originale.

```
bool  MathLog(
   double&         array[],     // array dei valori
   const double    base         // base del logaritmo
  );
```

Parametri

array[]

[in] Array di valori.

base

[in] La base del logaritmo.

array[]

[out] Array di valori di uscita.

result[]

[out] Array di valori di output.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathLog2

Calcola il logaritmo in base 2 per gli elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathLog2(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathLog2(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathLog10

Calcola il logaritmo in base 10 per gli elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathLog10(
   const double&   array[],   // array di valori
   double&         result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathLog10(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathLog1p

Calcola i valori del log(1+x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathLog1p(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathLog1p(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di uscita.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathDifference

Genera un array con differenze elemento di y[i]=x[i+lag]-x[i].

Versione per la generazione singola di un array di valori reali:

```
bool  MathDifference(
   const double  &array[],     // array di valori
   const int     lag,          // lag
   double        &result[]     // array di risultati
  );
```

Versione per la generazione singola di array di valori interi:

```
bool  MathDifference(
   const int     &array[],     // array di valori
   const int     lag,          // lag
   int           &result[]     // array dei risultati
  );
```

Versione per la generazione iterata di un'array di valori reali (il numero di iterazioni è impostato nei parametri di input):

```
bool  MathDifference(
   const double  &array[],     // array di valori
   const int     lag,          // lag
   const int     differences,  // numero di iterazioni
   double        &result[]     // array dei risultati
  );
```

Versione per la generazione iterata di un array di valori interi (il numero di iterazioni è impostato nei parametri di input):

```
bool  MathDifference(
   const int&    array[],      // array dei valori
   const int     lag,          // lag
   const int     differences,  // numero di iterazioni
   int&          result[]      // array dei risultati
  );
```

Parametri

array[]

[in] Array di valori.

lag

[in] Parametro Lag.

differences

[in] Il numero di iterazioni.

result[]

[out] Array per dare in output i risultati.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathSample

Genera un campione casuale dagli elementi dell'array.

Versione per lavorare con una serie di valori reali:

```
bool  MathSample(
   const double&  array[],          // array di valori
   const int      count,            // conteggio
   double&        result[]          // array dei risultati
  );
```

Versione per lavorare con un array di valori interi:

```
bool  MathSample(
   const int&     array[],          // array di valori
   const int      count,            // conteggio
   int&           result[]          // array dei risultati
  );
```

Versione per lavorare con un array di valori reali. È possibile avere un campione con sostituzione:

```
bool  MathSample(
   const double&  array[],          // array di valori
   const int      count,            // conteggio
   const bool     replace,          // flag
   double&        result[],         // array dei risultati
  );
```

Versione per lavorare con un'array di valori integer. È possibile avere un campione con sostituzione:

```
bool  MathSample(
   const int&     array[],          // array di valori
   const int      count,            // conteggio
   const bool     replace,          // flag
   int&           result[]          // array dei risultati
  );
```

Versione per lavorare con un array di valori reali, per il quale sono definite le probabilità di campionamento.

```
bool  MathSample(
   const double&  array[],          // array di valori
   double&        probabilities[],  // array di probabilità
   const int      count,            // conteggio
   double&        result[]          // array dei risultati
  );
```

Versione per lavorare con un array di valori integer, per il quale sono definite le probabilità di campionamento.

```
bool  MathSample(
   const int&     array[],           // array di valori
   double&        probabilities[],   // array di probabilità
   const int      count,             // conteggio
   int&           result[]           // array di risultati
  );
```

Versione per lavorare con un array di valori reali, per il quale sono definite le probabilità di campionamento. È possibile avere un campione con sostituzione:

```
bool  MathSample(
   const double&  array[],           // array di valori
   double&        probabilities[],   // array di probabilità
   const int      count,             // conteggio
   const bool     replace,           // flag
   double&        result[]           // array dei risultati
  );
```

Versione per lavorare con un array di valori integer, per il quale sono definite le probabilità di campionamento. È possibile avere un campione con sostituzione:

```
bool  MathSample(
   const int&     array[],           // array di valori
   double&        probabilities[],   // array di probabilità
   const int      count,             // conteggio
   const bool     replace,           // flag
   int&           result[]           // array di risultati
  );
```

Parametri

array[]

[in] Array di valori integer.

probabilities[]

[in] Array delle probabilità per il campionamento degli elementi.

count

[in] Il numero degli elementi.

replace

[in] Parametro che consente il campionamento con sostituzione.

result[]

[out] Array per dare in output i risultati.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Nota

L'argomento replace=true consente di eseguire il campionamento casuale degli elementi con sostituzione di nuovo alla sequenza originale.

MathTukeySummary

Calcola la sintesi a cinque numeri Tukey (minimo, quartile inferiore, mediano, quartile superiore, massimo) per gli elementi dell'array.

```
bool  MathTukeySummary(
   const double&  array[],       // array dei valori
   const bool     removeNAN,     // flag
   double&        minimum,       // valore minimo
   double&        lower_hinge,   // quartile inferiore
   double&        median,        //  valore mediano
   double&        upper_hinge,   // quartile superiore
   double&        maximum        // valore massimo
  );
```

Parametri

array[]

[in] Array dei valori reali.

removeNAN

[in] Flag che indica se i valori non numerici devono essere rimossi.

minimum

[out] Variabile per memorizzare il valore minimo.

lower_hinge

[out] Variabile per memorizzare il quartile inferiore.

median

[out] Variabile per memorizzare il valore mediano.

upper_hinge

[out] Variabile per memorizzare il quartile superiore.

maximum

[out] variabile per memorizzare il valore massimo.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathRange

Calcola i valori minimi e massimi di elementi array.

```
bool  MathRange(
   const double&  array[],  // array di valori
   double&        min,      // valore minimo
   double&        max       // valore massimo
  );
```

Parametri

array[]

[in] Array di valori.

min

[out] Variabile per memorizzare il valore minimo.

max

[out] variabile per memorizzare il valore massimo.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathMin

Restituisce il valore minimo di tutti gli elementi dell'array.

```
double  MathMin(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Il valore minimo.

MathMax

Restituisce il valore massimo di tutti gli elementi dell'array.

```
double  MathMax(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Il valore massimo.

MathSum

Restituisce la somma degli elementi dell'array.

```
double  MathSum(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

La somma degli elementi.

MathProduct

Restituisce il prodotto di elementi array.

```
double  MathProduct(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Il prodotto degli elementi.

MathStandardDeviation

Calcola la deviazione standard di elementi array.

```
double  MathStandardDeviation(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Deviazione standard.

MathAverageDeviation

The function calculates the average absolute deviation of array elements.

```
double  MathAverageDeviation(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

La deviazione media assoluta di elementi array.

MathMedian

Calcola il valore mediano di elementi dell'array.

```
double  MathMedian(
   double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Il valore mediano.

MathMean

Calcola i valori medi di elementi dell'array.

```
double  MathMean(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Il valore medio.

MathVariance

La funzione calcola la varianza (secondo momento) di elementi array.

```
double  MathVariance(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Valore della varianza.

MathSkewness

La funzione calcola l'indice di asimmetria (terzo momento) di elementi dell'array.

```
double  MathSkewness(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Asimmetria.

MathKurtosis

La funzione calcola la curtosi (quarto momento) di elementi array.

```
double  MathKurtosis(
   const double&  array[]   // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

Valore di ritorno

Curtosi.

MathExpm1

Calcola i valori della funzione exp(x)-1 di elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathExpm1(
   const double&   array[],   // array di valori
   double&         result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathExpm1(
   double&         array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di output.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathSinh

Calcola i valori della funzione sinh(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathSinh(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathSinh(
   double&       array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di output.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCosh

Calcola i valori della funzione cosh(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathCosh(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathCosh(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di output.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathTanh

Calcola i valori della funzione tanh(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathTanh(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathTanh(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di output.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathArcsinh

Calcola i valori della funzione arcsinh(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathArcsinh(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathArcsinh(
   double&         array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di output.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathArccosh

Calcola i valori della funzione arccosh(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathArccosh(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathArccosh(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di output.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathArctanh

Calcola i valori della funzione arctanh(x) per elementi dell'array.

Versione con output dei risultati in un nuovo array:

```
bool  MathArctanh(
   const double&  array[],   // array di valori
   double&        result[]   // array di risultati
  );
```

Versione con output dei risultati nell'array originale:

```
bool  MathArctanh(
   double&        array[]    // array di valori
  );
```

Parametri

array[]

[in] Array di valori.

result[]

[out] Array di valori di output.

array[]

[out] Array di valori di uscita.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathSignif

Arrotonda un valore per il numero specificato di cifre nella mantissa.

Versione per lavorare con un valore reale:

```
double  MathSignif(
   const double   x,          // valore
   const int      digits      // numero di posizioni decimali
  );
```

Valore di ritorno

Il valore arrotondato.

Versione per lavorare con un array di valori reali e con l'output dei risultati in un array separato:

```
bool  MathSignif(
   const double&  array[],    // array di valori
   int            digits,     // numero di posizioni decimali
   double         result[]    // array dei risultati
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Versione per lavorare con un array di valori reali e con l'output dei risultati all'array originale:

```
bool  MathSignif(
   double&        array[],    // array di valori
   int            digits      // numero di posizioni decimali
  );
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Parametri

x

[in] il valore reale da arrotondare.

digits

[in] Numero di cifre decimali.

array[]

[in] Array di valori reali.

array[]

[out] Array di valori di uscita.

result[]

[out] Array di valori di uscita.

MathRank

Calcola le fila di elementi dell'array.

Versione per lavorare con una serie di valori reali:

```
bool  MathRank(
   const double&  array[],  // array di valori
   double&        rank[]    // array di ranghi
  );
```

Versione per lavorare con un array di valori interi:

```
bool  MathRank(
   const int&     array[],  // array di valori
   double&        rank[]    // array di ranghi
  );
```

Parametri

array[]

[in] Array di valori.

rank[]

[out] Array dove dare in output i ranghi.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCorrelationPearson

Calcola il coefficiente di correlazione di Pearson.

Versione per lavorare con array di valori reali:

```
bool  MathCorrelationPearson(
   const double&  array1[],  // il primo array di valori
   const double&  array2[],  // il secondo array di valori
   double&        r          // coefficiente di correlazione
  );
```

Versione per lavorare con array di valori interi:

```
bool  MathCorrelationPearson(
   const int&     array1[],  // il primo array di valori
   const int&     array2[],  // il secondo array di valori
   double&        r          // coefficiente di correlazione
  );
```

Parametri

array1[]

[in] Il primo array di valori.

array2[]

[in] Il secondo array di valori.

r

[out] variabile per memorizzare il coefficiente di correlazione.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCorrelationSpearman

Calcola il coefficiente di correlazione di Spearman.

Versione per lavorare con array di valori reali:

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // il primo array di valori
   const double&  array2[],  // il secondo array di valori
   double&        r          // coefficiente di correlazione
  );
```

Versione per lavorare con array di valori interi:

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // il primo array di valori
   const int&     array2[],   // il secondo array di valori
   double&        r           // coefficiente di correlazione
  );
```

Parametri

array1[]

[in] Il primo array di valori.

array2[]

[in] Il secondo array di valori.

r

[out] variabile per memorizzare il coefficiente di correlazione.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCorrelationKendall

Calcola il coefficiente di correlazione di Kendall.

Versione per lavorare con array di valori reali:

```
bool  MathCorrelationKendall(
   const double&  array1[],  // il primo array di valori
   const double&  array2[],  // il secondo array di valori
   double&        tau        // coefficiente di correlazione
  );
```

Versione per lavorare con array di valori interi:

```
bool  MathCorrelationKendall(
   const int&     array1[],  // il primo array di valori
   const int&     array2[],  // il secondo array di valori
   double&        tau        // coefficiente di correlazione
  );
```

Parametri

array1[]

[in] Il primo array di valori.

array2[]

[in] Il secondo array di valori.

tau

[out] variabile per memorizzare il coefficiente di correlazione.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathQuantile

Calcola quantili campione corrispondenti alle probabilità determinate: Q[i](p) = (1 - gamma) * x[j] + gamma *x[j+1]

```
bool  MathQuantile(
   const double&  array[],     // array di valori
   const double&  probs[],     // array delle probabilità
   double&        quantile[]   // array per l'output dei quantili
  );
```

Parametri

array[]

[in] Array di valori.

probs[]

[in] Array delle probabilità.

quantile[]

[out] Array per l' output dei quantili.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathProbabilityDensityEmpirical

La funzione calcola la funzione di densità di probabilità empirica (pdf) per valori casuali da un array.

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // array odi valori random
   const int      count,     // il numero di coppie
   double&        x[],       // array di valori x
   double&        pdf[]      // array di valori pdf
  );
```

Parametri

array[]

[in] Array di valori casuali.

count

[in] Il numero di coppie (x, pdf(x)).

x[]

[out] Array per l'output di valori x .

pdf[]

[out] Array per emettere i valori pdf(x).

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

MathCumulativeDistributionEmpirical

La funzione calcola la funzione empirica cumulata di distribuzione (cdf) per valori casuali da un'array.

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],    // array odi valori random
   const int      count,      // il numero di coppie
   double&        x[],        // array di valori x
   double&        cdf[]       // array di valori cdf
  );
```

Parametri

array[]

[in] Array di valori casuali.

count

[in] Il numero di (x, CDF(x)) coppie.

x[]

[out] Array per l'output di valori x .

cdf[]

[out] Array per l'output di valori cdf(x) .

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Fuzzy è una libreria per lavorare con la logica fuzzy

La logica fuzzy è una sintesi della logica Aristotelica tradizionale quando la verità viene contrassegnata come variabile linguistica. La logica fuzzy, equivalente alla logica classica, ha le proprie operazioni logiche Fuzzy su insiemi fuzzy definiti. Ci sono le stesse operazioni per insiemi(sets) fuzzy così come per insiemi ordinari, solo, il loro calcolo è di gran lunga più difficile. Dobbiamo anche notare che la composizione degli insiemi fuzzy si costituisce come un insieme fuzzy.

I principi fondamentali della logica fuzzy, che lo distingue dalla logica classica sono di massima vicinanza alla riflessione della realtà e ad un alto livello di soggettività, che può portare ad errori significativi nei calcoli.

Modello Fuzzy (o sistema) è un modello matematico il cui calcolo è basato su logica fuzzy. La costruzione di tali modelli è applicabile quando l'oggetto di studio ha una formalizzazione debole e la sua esatto descrizione matematica è troppo complessa o sconosciuta. La qualità dei valori di output di questi modelli (errore modello) dipende direttamente dall' Expert Advisor, che ha istituito questo modello. L'opzione migliore per ridurre al minimo gli errori è quella di disegnare il modello più completo ed esaustivo e successivamente regolarlo con apprendimento automatico su un ampio insieme di addestramento.

Il progresso di una costruzione del modello può essere suddiviso in tre fasi principali:

1. Definizione delle caratteristiche di input ed output di un modello.
2. Costruire una base di conoscenze.
3. La selezione di uno dei metodi di inferenza Fuzzy (Mamdani o Sugeno).

La prima fase effettua direttamente le conseguenti due e determina il futuro funzionamento del modello. Una base di conoscenza o, come a volte chiamato, regola base, è un insieme di regole fuzzy tipo: "se, allora"(if, then) che definiscono il rapporto tra input e output dell'oggetto esaminato. Il numero di regole nel sistema non è limitato ed è determinato anche dall' Expert Advisor. Il formato generalizzata di regole fuzzy è il seguente:

Se vo è la condizione della regola, allora vi è conclusione della regola.

ça condizione della regola descrive lo stato corrente dell'oggetto, e la conclusione regola conclusione — come questa condizione colpisce l'oggetto. Vista generale delle condizioni e le conclusioni non possono essere selezionate perché sono determinate da un'inferenza fuzzy.

Ogni regola nel sistema ha il suo peso — questa caratteristica definisce l'importanza di una regola nel modello. Fattori di ponderazione(pesatura) sono assegnati ad una regola nel range [0, 1]. In molti esempi con modelli fuzzy, che possono essere trovati in letteratura, i dati di peso non vengono specificati, ma non significa che non siano presenti. Infatti, in tal caso per ogni regola dal database, il peso è fisso e pari all' unità. Ci possono essere due tipi di termini e conclusioni per ogni regola:

1. semplice — include una variabile fuzzy;
2. complesso — include diverse variabili fuzzy

A seconda della base di conoscenza creata, il sistema di inferenza fuzzy viene determinato per un modello. L'inferenza logica fuzzy è una ricevuta di conclusione in forma di un insieme fuzzy corrispondente al valore attuale degli ingressi con uso di base di conoscenza ed operazioni fuzzy. I due tipi principali di inferenza Fuzzy sono Mamdani e Sugeno.

Funzioni di appartenenza

Una **funzione di appartenenza** è una funzione che permette di calcolare il grado di appartenenza di un elemento casuale del set universale per un insieme fuzzy. Di conseguenza, il dominio di una funzione di appartenenza dovrebbe essere compreso nell'intervallo [0, 1].

Nella maggior parte dei casi, la funzione di appartenenza è continua e monotona.

Classi di funzioni di appartenenza	Descrizione
CConstantMembershipFunction	Classe per implementare una funzione di appartenenza come una linea retta parallela con l'asse coordinate
CCompositeMembershipFunction	Classe per l'attuazione di una composizione di funzioni di appartenenza
CDifferencTwoSigmoidalMembershipFunction	Classe per implementare la funzione di appartenenza in forma di una differenza tra due funzioni sigmoide con i parametri A1, A2, C1 e C2
CGeneralizedBellShapedMembershipFunction	Classe per implementare una funzione di appartenenza a campana generalizzata con i parametri A, B e C
CNormalCombinationMembershipFunction	Classe per implementare una funzione di appartenenza Gaussiana a due facce con i parametri B1, B2, Sigma1 e Sigma2
CNormalMembershipFunction	Classe per implementare una funzione di appartenenza gaussiana simmetrica con i parametri B e Sigma
CP_ShapedMembershipFunction	Classe per implementare una funzione di appartenenza di forma-pi con i parametri A, B, C e D
CProductTwoSigmoidalMembershipFunction	Classe per implementare la funzione di appartenenza in forma di un prodotto di due funzioni sigmoide con i parametri A1, A2, C1 e C2
CS_ShapedMembershipFunction	Classe per implementare una funzione di appartenenza stile-S con i parametri Á e B
CTrapezoidMembershipFunction	Classe per implementare una funzione di appartenenza di forma trapezoidale con i parametri X1, X2, X3 e X4
CTriangularMembershipFunction	Classe per implementare una funzione di appartenenza triangolo con i parametri X1, X2 e X3

Classi di funzioni di appartenenza	Descrizione
CSigmoidalMembershipFunction	Classe per implementare una funzione di appartenenza sigmoide con i parametri A e C
CZ_ShapedMembershipFunction	Classe per implementare una funzione di appartenenza stile-z ccon i parametri A e B.
IMembershipFunction	Classe di base per tutte le classi di funzione di appartenenza.

CConstantMembershipFunction

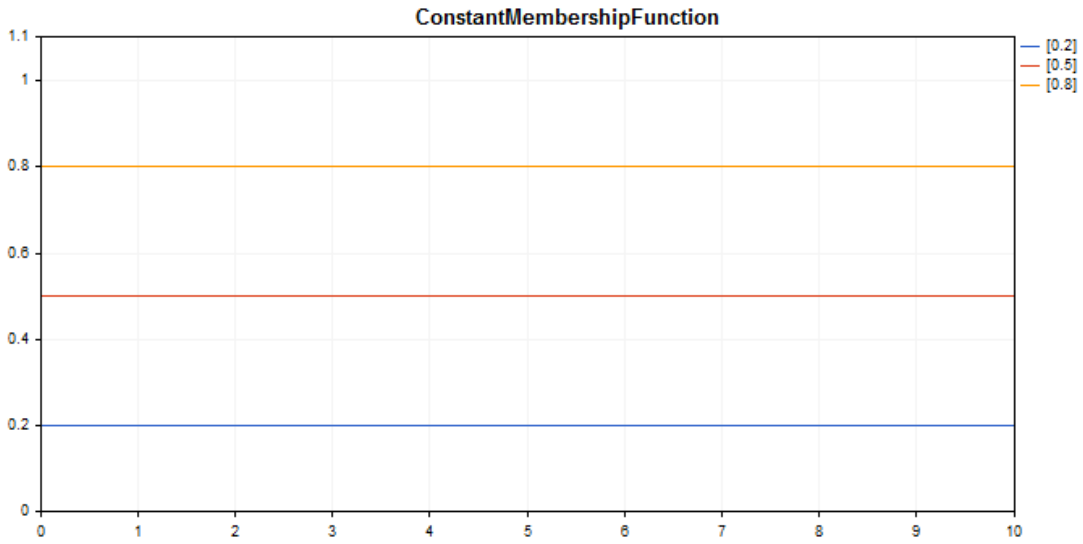
Classe per implementare una funzione di appartenenza come una linea retta in parallelo con l'asse delle coordinate.

Descrizione

La funzione è descritta dall'equazione:

y(x)=c

Pertanto, il grado di appartenenza per la funzione è lo stesso lungo tutto l'asse numerico ed è uguale al parametro specificato nel costruttore.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CConstantMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
IMembershipFunction
CConstantMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                  ConstantMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CConstantMembershipFunction func1(0.2);
CConstantMembershipFunction func2(0.5);
CConstantMembershipFunction func3(0.8);
//--- Crea wrapper per funzioni di appartenenza
double ConstantMembershipFunction1(double x) { return(func1.GetValue(x)); }
double ConstantMembershipFunction2(double x) { return(func2.GetValue(x)); }
double ConstantMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"ConstantMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"ConstantMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ConstantMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(ConstantMembershipFunction1,0.0,10.0,1.0,CURVE_LINES,"[0.2]");
   graphic.CurveAdd(ConstantMembershipFunction2,0.0,10.0,1.0,CURVE_LINES,"[0.5]");
   graphic.CurveAdd(ConstantMembershipFunction3,0.0,10.0,1.0,CURVE_LINES,"[0.8]");
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
```

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const double  x      // membership function argument
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

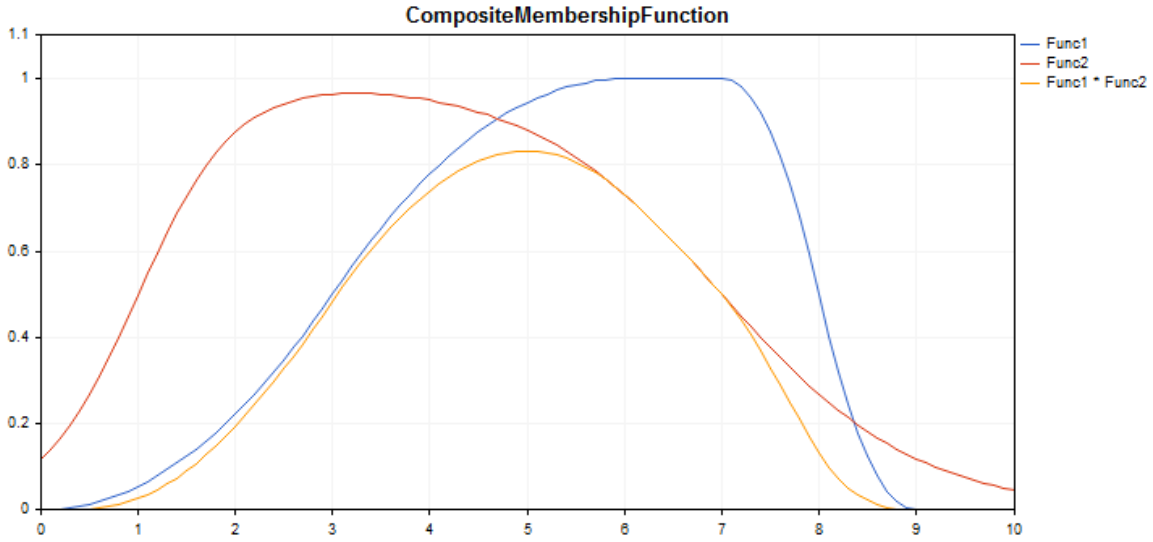
Valore della funzione di appartenenza.

CCompositeMembershipFunction

Classe per l'implementazione di una composizione di funzioni di appartenenza.

Descrizione

La composizione di funzioni di appartenenza è una combinazione di due o più funzioni di appartenenza utilizzando un operatore specificato.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
IMembershipFunction
CCompositeMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
CompositionType	Imposta l'operatore composizione.
MembershipFunctions	Ottiene l'elenco delle funzioni di appartenenza.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CP_ShapedMembershipFunction func2(0,6,7,9);
CCompositeMembershipFunction composite(ProdMF,GetPointer(func1),GetPointer(func2));
//--- Crea wrapper per funzioni di appartenenza
double ProductTwoSigmoidalMembershipFunctions(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction(double x) { return(func2.GetValue(x)); }
double CompositeMembershipFunction(double x) { return(composite.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"CompositeMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"CompositeMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("CompositeMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(P_ShapedMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions,0.0,10.0,0.1,CURVE_LINES,"F
   graphic.CurveAdd(CompositeMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1 * Func
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
```

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

CompositionType

Imposta l'operatore composizione.

```
void  CompositionType(
   MfCompositionType  value      // tipo operatore
   )
```

Parametri

valore

[in] Tipo di operatore Composizione.

Nota

I seguenti tipi di operatore sono disponibili:

- MinMF (minimo di funzioni)
- MaxMF (massimo di funzioni)
- ProdMF (prodotto di funzioni)
- SumMF (somma di funzioni)

MembershipFunctions

Ottiene l'elenco delle funzioni di appartenenza inclusi in una composizione.

```
CList*  MembershipFunctions(
   void        // elenco di funzioni di appartenenza
   )
```

Valore di ritorno

Elenco di funzioni di appartenenza.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

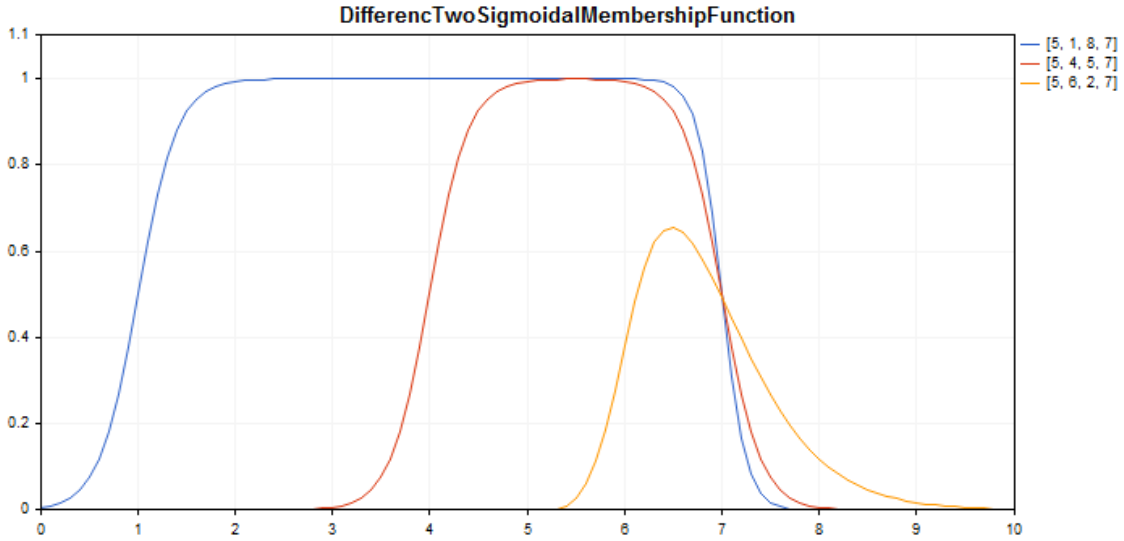
Valore della funzione di appartenenza.

CDifferencTwoSigmoidalMembershipFunction

Classe per implementare la funzione di appartenenza in forma di una differenza tra due funzioni sigmoide con i parametri A1, A2, C1 e C2.

Descrizione

La funzione si basa su una curva sigmoidale. Essa consente la creazione di funzioni di appartenenza dei valori pari ad 1 che iniziano con un valore di argomento. Tali funzioni sono adatte se è necessario impostare tali termini linguistici come "short" o "long".



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject

IMembershipFunction

CDifferencTwoSigmoidalMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
A1	Ottiene ed imposta il primo rapporto di pendenza della funzione di appartenenza.

I metodi della classe	Descrizione
A2	Ottiene ed imposta il secondo rapporto di pendenza della funzione di appartenenza.
C1	Ottiene ed imposta il primo parametro della coordinata inflession della funzione di appartenenza.
C2	Ottiene ed imposta il secondo parametro della coordinata inflession della funzione di appartenenza.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CDifferencTwoSigmoidalMembershipFunction func1(5,1,8,7);
CDifferencTwoSigmoidalMembershipFunction func2(5,4,5,7);
CDifferencTwoSigmoidalMembershipFunction func3(5,6,2,7);
//--- Crea wrapper per funzioni di appartenenza
double DifferencTwoSigmoidalMembershipFunction1(double x) { return(func1.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction2(double x) { return(func2.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"DifferencTwoSigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"DifferencTwoSigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("DifferencTwoSigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
```

```
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (metodo Get)

Ottiene il primo rapporto di pendenza della funzione di appartenenza.

```
double  A1()
```

Valore di ritorno

Valore del rapporto di pendenza.

A1 (metodo Set)

Imposta il primo rapporto di pendenza della funzione di appartenenza.

```
void  A1(
   const double  a1      // valore rapporto di pendenza
   )
```

Parametri

a1

[in] Valore rapporto di pendenza.

A2 (metodo Get)

Ottiene il secondo rapporto di pendenza della funzione di appartenenza.

```
double  A2()
```

Valore di ritorno

Valore del rapporto di pendenza.

A2 (metodo Set)

Imposta il secondo rapporto di pendenza della funzione di appartenenza.

```
void  A2(
   const double  a2      // valore rapporto di pendenza
   )
```

Parametri

a2

[in] Valore rapporto di pendenza.

C1 (metodo Get)

Ottiene il primo parametro della coordinata inflessione della funzione di appartenenza.

```
double  C1()
```

Valore di ritorno

Valore corrdinata inflessione.

C1 (metodo Set)

Imposta il primo parametro della coordinata inflessione della funzione di appartenenza.

```
void  C1(
   const double  c1      // valore corrdinata inflessione
   )
```

Parametri

c1

[in] Valore corrdinata inflessione.

C2 (metodo Get)

Ottiene il secondo parametro della coordinata inflessione della funzione di appartenenza.

```
double  C2()
```

Valore di ritorno

Valore corrdinata inflessione.

C2 (metodo Set)

Imposta il secondo parametro della coordinata inflessione della funzione di appartenenza.

```
void  C2(
   const double  c2      // valore corrdinata inflessione
   )
```

Parametri

c2

[in] Valore corrdinata inflessione.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const double  x      // argomento funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

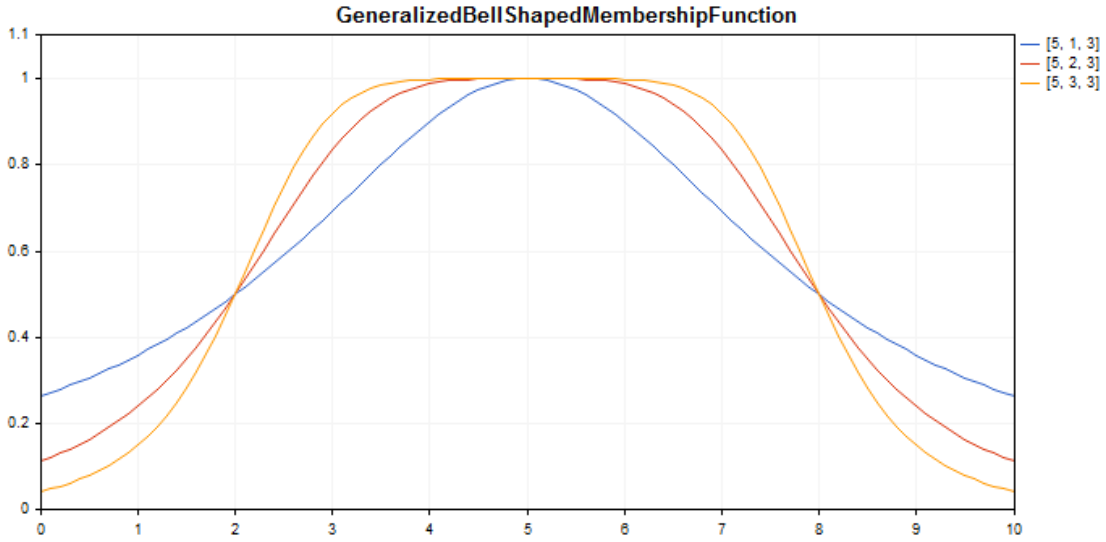
Valore della funzione di appartenenza.

CGeneralizedBellShapedMembershipFunction

Classe per implementare una funzione di appartenenza a forma-di-campana generalizzata con i parametri A, B e C.

Descrizione

Dunzione di appartenenza a a foma-di-campana generalizzata è simile a funzioni Gaussiane. La funzione è smussata ed assume valori diversi da zero lungo tutta l'area di definizione.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
IMembershipFunction
CGeneralizedBellShapedMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
A	Ottiene ed imposta il rapporto di concentrazione funzione di appartenenza.
B	Ottiene ed imposta il rapporto pendenza funzione di appartenenza.

I metodi della classe	Descrizione
C	Ottiene ed imposta la funzione di appartenenza massime coordinate.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CGeneralizedBellShapedMembershipFunction func1(5, 1, 3);
CGeneralizedBellShapedMembershipFunction func2(5, 2, 3);
CGeneralizedBellShapedMembershipFunction func3(5, 3, 3);
//--- Crea wrapper per funzioni di appartenenza
double GeneralizedBellShapedMembershipFunction1(double x) { return(func1.GetValue(x));
double GeneralizedBellShapedMembershipFunction2(double x) { return(func2.GetValue(x));
double GeneralizedBellShapedMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"GeneralizedBellShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"GeneralizedBellShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("GeneralizedBellShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (metodo Get)

Ottiene il rapporto di concentrazione funzione di appartenenza.

```
double  A()
```

Valore di ritorno

Valore rapporto di concentrazione.

A (metodo Set)

Imposta il rapporto di concentrazione funzione di appartenenza.

```
void  A(
   const double  a      // valore rapporto di concentrazione
   )
```

Parametri

a

[in] Rapporto di concentrazione funzione di appartenenza.

B (metodo Get)

Ottiene il rapporto pendenza funzione di appartenenza.

```
double  B()
```

Valore di ritorno

Valore del rapporto di pendenza.

B (metodo Set)

Imposta il rapporto pendenza funzione di appartenenza.

```
void  B(
   const double  b      // valore rapporto di pendenza
   )
```

Parametri

b

[in] Rapporto di pendenza della funzione di appartenenza.

C (metodo Get)

Ottiene la coordinata massima della funzione di appartenenza.

```
double  C()
```

Valore di ritorno

Coordinata massima funzione di appartenenza.

C (metodo Set)

Imposta la coordinata massima funzione di appartenenza.

```
void  C(
   const double  c      // valore massima coordinata
   )
```

Parametri

c

[in] Coordinata massima funzione di appartenenza.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

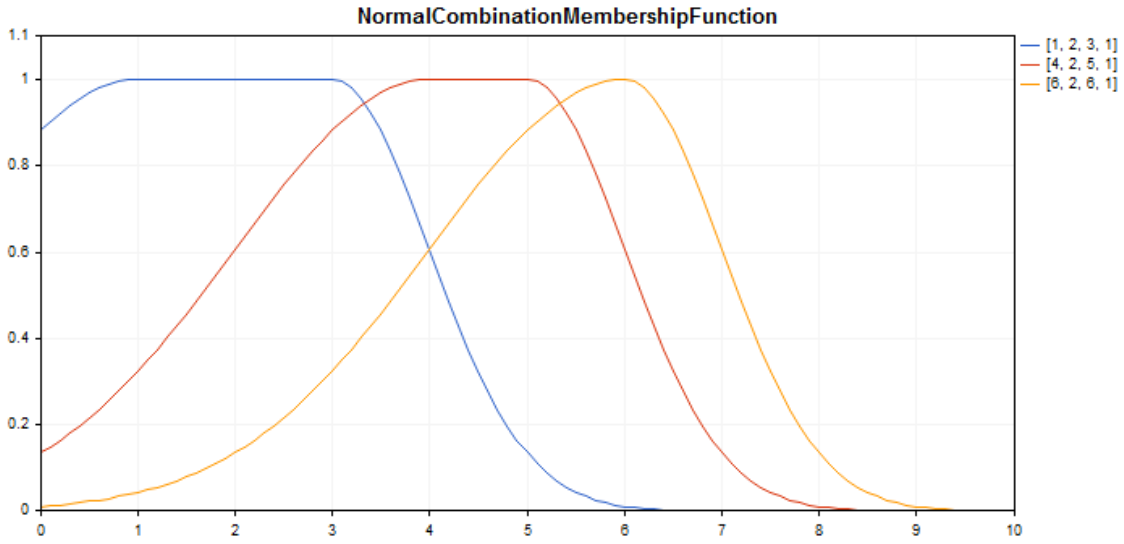
Valore della funzione di appartenenza.

CNormalCombinationMembershipFunction

Classe per implementare una funzione di appartenenza gaussiana a due facce con i parametri B1, B2, Sigma1 e Sigma2.

Descrizione

La funzione di appartenenza gaussiana a due facce (lati) è formata utilizzando la distribuzione gaussiana. Essa consente di impostare funzioni di appartenenza asimmetriche. La funzione è smussata ed assume valori diversi da zero lungo tutta l'area di definizione.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
IMembershipFunction
CNormalCombinationMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
B1	Ottiene ed imposta il valore del primo centro di funzione di appartenenza.

I metodi della classe	Descrizione
B2	Ottiene ed imposta il valore del secondo centro di funzione di appartenenza.
Sigma1	Ottiene ed imposta il primo parametro della curvatura della funzione di appartenenza.
Sigma2	Ottiene ed imposta il secondo parametro della curvatura della funzione di appartenenza.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                       Copyright 2016, MetaQuotes Software Corp.  |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CNormalCombinationMembershipFunction func1(1,2,3,1);
CNormalCombinationMembershipFunction func2(4,2,5,1);
CNormalCombinationMembershipFunction func3(6,2,6,1);
//--- Crea wrapper per funzioni di appartenenza
double NormalCombinationMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalCombinationMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalCombinationMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"NormalCombinationMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalCombinationMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalCombinationMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(NormalCombinationMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[1,
```

```
   graphic.CurveAdd(NormalCombinationMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[4,
   graphic.CurveAdd(NormalCombinationMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[6,
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B1 (metodo Get)

Ottiene il valore del primo centro della funzione di appartenenza.

```
double  B1()
```

Valore di ritorno

Valore del primo centro della funzione di appartenenza.

B1 (metodo Set)

Imposta il valore del primo centro della funzione di appartenenza.

```
void  B1(
   const double  b1      // valore del primo centro
   )
```

Parametri

b

[in] Valore del primo centro della funzione di appartenenza.

B2 (metodo Get)

Ottiene il valore del secondo centro della funzione di appartenenza.

```
double  B2()
```

Valore di ritorno

Il valore del secondo centro della funzione di appartenenza.

B2 (metoto Set)

Imposta il valore del secondo centro della funzione di appartenenza.

```
void  B2(
   const double  b2      // valore del secondo centro
   )
```

Parametri

b2

[in] Valore del secondo centro della funzione di appartenenza.

Sigma1 (metodo Get)

Ottiene il primo parametro della curvatura della funzione di appartenenza.

```
double  Sigma1()
```

Valore di ritorno

Il valore del primo parametro della curvatura della funzione di appartenenza.

Sigma1 (metodo Set)

Imposta il valore del primo parametro della curvatura della funzione di appartenenza.

```
void  Sigma1(
   const double  sigma1      // valore del parametro prima curvatura
   )
```

Parametri

sigma1

[in] Il primo parametro della curvatura della funzione di appartenenza.

Sigma2 (metoto Get)

Ottiene il secondo parametro della curvatura della funzione di appartenenza.

```
double  Sigma2()
```

Valore di ritorno

Il valore del secondo parametro della curvatura della funzione di appartenenza.

Sigma2 (metodo Set)

Imposta il valore del secondo parametro della curvatura della funzione di appartenenza.

```
void  Sigma2(
   const double  sigma2      // valore del parametro seconda curvatura
   )
```

Parametri

sigma2

[in] Il secondo parametro curvatura della funzione di appartenenza.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

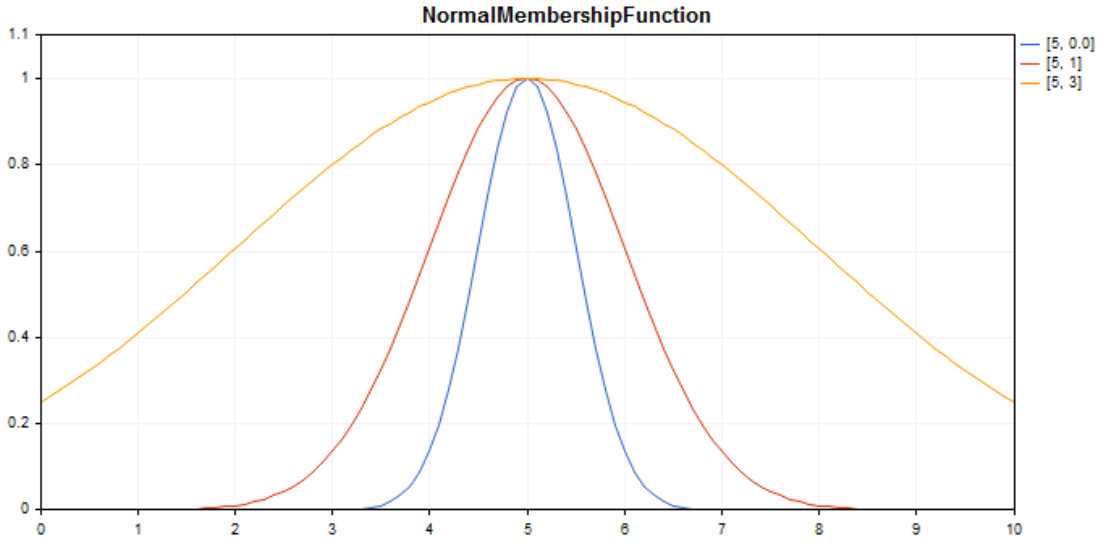
Valore della funzione di appartenenza.

CNormalMembershipFunction

Classe per implementare una funzione di appartenenza simmetrica gaussiana con i parametri B e Sigma.

Descrizione

La funzione di appartenenza gaussiana simmetrica è formata utilizzando la distribuzione gaussiana. La funzione è smussata ed assume valori diversi da zero lungo tutta l'area di definizione.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CNormalMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
IMembershipFunction
CNormalMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
B	Ottiene ed imposta il centro della funzione di appartenenza.
Sigma	Ottiene ed imposta il parametro della curvatura della funzione di appartenenza.

I metodi della classe	Descrizione
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                     NormalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CNormalMembershipFunction func1(5,0.5);
CNormalMembershipFunction func2(5,1);
CNormalMembershipFunction func3(5,3);
//--- Crea wrapper per funzioni di appartenenza
double NormalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"NormalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(NormalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[5, 0.0]");
   graphic.CurveAdd(NormalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[5, 1]");
   graphic.CurveAdd(NormalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[5, 3]");
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
```

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B (metodo Get)

Ottiene il centro della funzione di appartenenza.

```
double  B()
```

Valore di ritorno

Il valore del centro della funzione di appartenenza.

B (metodo Set)

Imposta il valore del centro della funzione di appartenenza.

```
void  B(
   const double  b      // valore del centro della funzione
   )
```

Parametri

b

[in] Valore del centro della funzione di appartenenza.

Sigma (metodo Get)

Ottiene il parametro della curvatura della funzione di appartenenza.

```
double  Sigma()
```

Valore di ritorno

Il valore del parametro curvatura della funzione di appartenenza.

Sigma (metodo Set)

Imposta il valore del parametro curvatura della funzione di appartenenza.

```
void  Sigma(
   const double  sigma      // valore parametro curvatura
   )
```

Parametri

sigma

[in] Parametro curvatura della funzione di appartenenza.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const double  x      // argomento
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

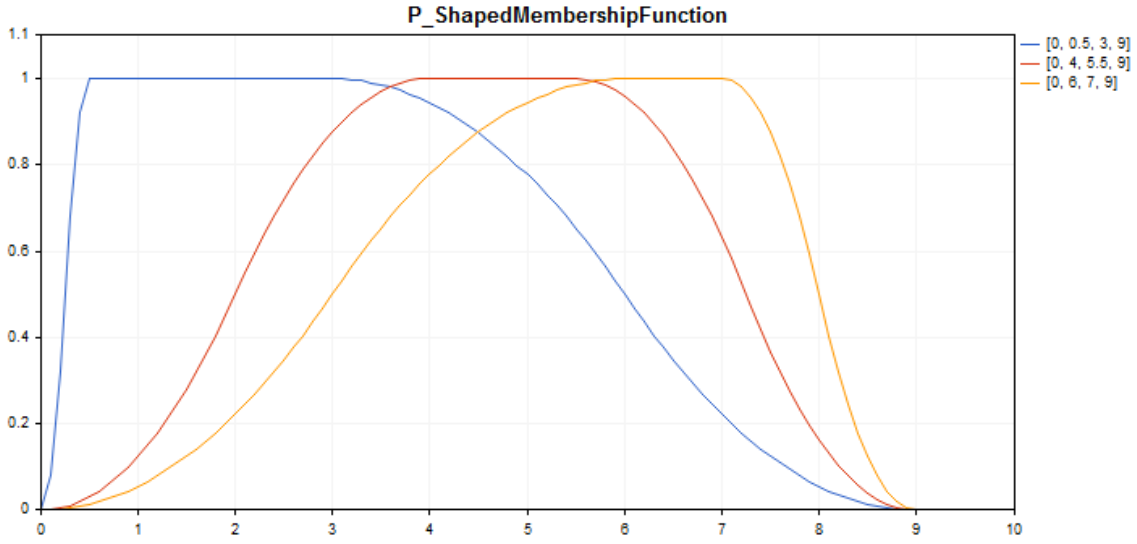
Valore della funzione di appartenenza.

CP_ShapedMembershipFunction

Classe per implementare una funzione di appartenenza a forma-pi con i parametri A, B, C e D.

Descrizione

La funzione di appartenenza a forma di pi ha la forma di un trapezio curvilineo. La funzione permette di impostare funzioni di appartenenza asimmetriche con una transizione graduale dalla valutazione pessimistia ad ottimista del numero fuzzy.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
  IMembershipFunction
    CP_ShapedMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
A	Ottiene ed imposta il parametro dell'inizio del set fuzzy.
B	Ottiene ed imposta il primo parametro del nucleo dell' insieme fuzzy.

I metodi della classe	Descrizione
C	Ottiene ed imposta il secondo parametro del nucleo dell' insieme fuzzy.
D	Ottiene e imposta il parametro di fine del set fuzzy.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                    P_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CP_ShapedMembershipFunction func1(0,0.5,3,9);
CP_ShapedMembershipFunction func2(0,4,5.5,9);
CP_ShapedMembershipFunction func3(0,6,7,9);
//--- Crea wrapper per funzioni di appartenenza
double P_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double P_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"P_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"P_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("P_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(P_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 0.5, 3,
   graphic.CurveAdd(P_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 4, 5.5,
   graphic.CurveAdd(P_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[0, 6, 7, 9
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
```

```
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (metodo Get)

Ottiene il parametro inizio del set fuzzy.

```
double  A()
```

Valore di ritorno

Il parametro di inizio del set fuzzy.

A (metodo Set)

Imposta il parametro di inizio del set fuzzy.

```
void  A(
   const double  a      // parametro di inizio del set fuzzy
   )
```

Parametri

a

[In] Parametro dell'inizo fuzzy set.

B (metodo Get)

Ottiene il primo parametro del core del set fuzzy.

```
double  B()
```

Valore di ritorno

Il primo parametro del nucleo del set fuzzy.

B (metodo Set)

Imposta il primo parametro del nucleo del set fuzzy.

```
void  B(
   const double  b      // valore del primo parametro del core del set fuzzy
   )
```

Parametri

b

[in] Il primo parametro del nucleo del set fuzzy.

C (metodo Get)

Ottiene il secondo parametro del nucleo del set fuzzy.

```
double  C()
```

Valore di ritorno

Il secondo parametro del nucleo del set fuzzy.

C (metodo Set)

Imposta il secondo parametro del nucleo del set fuzzy.

```
void  C(
   const double  c      // valore del secondo parametro del nucleo del set fuzzy
   )
```

Parametri

c

[in] Il secondo parametro del nucleo del set fuzzy.

D (metodo Get)

Ottiene il parametro della fine del set fuzzy.

```
double  D()
```

Valore di ritorno

Valore del parametro della fine del set fuzzy.

D (metodo Set)

Imposta il parametro della fine del set fuzzy.

```
void  D(
   const double  d      // valore del parametro della fine del set fuzzy
   )
```

Parametri

d

[in] Valore del parametro della fine del set fuzzy.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const double  x
   )
```

Parametri

x

[in] Argomento funzione di appartenenza

Valore di ritorno

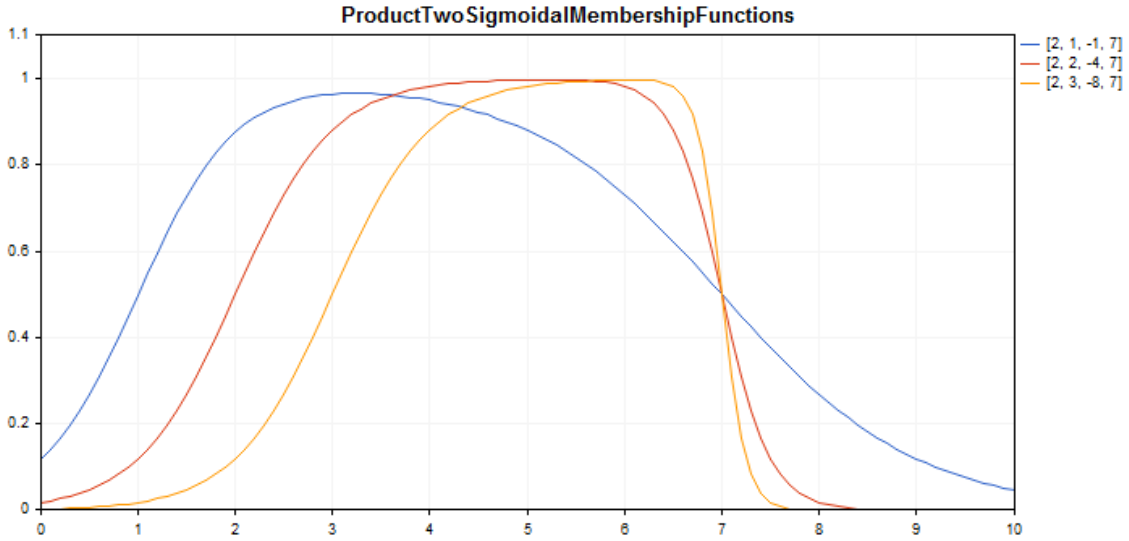
Valore della funzione di appartenenza

CProductTwoSigmoidalMembershipFunction

Classe per implementare la funzione di appartenenza in forma di un prodotto di due funzioni sigmoidi con i parametri A1, A2, C1 e C2.

Descrizione

Un prodotto di due funzioni di appartenenza sigmoide è applicato per impostare le funzioni asimmetriche smussate. Esso consente la creazione di funzioni di appartenenza dei valori pari ad 1 che iniziano con un valore di argomento. Tali funzioni sono adatte se è necessario impostare tali termini linguistici come "short" o "long".



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject

IMembershipFunction

CProductTwoSigmoidalMembershipFunctions

I metodi della classe

I metodi della classe	Descrizione
A1	Ottiene ed imposta il primo rapporto di pendenza della funzione di appartenenza.

I metodi della classe	Descrizione
A2	Ottiene ed imposta il secondo rapporto di pendenza della funzione di appartenenza.
C1	Ottiene il primo parametro della coordinata inflessione della funzione di appartenenza.
C2	Ottiene il secondo parametro della coordinata inflessione della funzione di appartenenza.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.mq5 |
//|                       Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CProductTwoSigmoidalMembershipFunctions func2(2,2,-4,7);
CProductTwoSigmoidalMembershipFunctions func3(2,3,-8,7);
//--- Crea wrapper per funzioni di appartenenza
double ProductTwoSigmoidalMembershipFunctions1(double x) { return(func1.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions2(double x) { return(func2.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"ProductTwoSigmoidalMembershipFunctions",0,30,30,780,380))
     {
      graphic.Attach(0,"ProductTwoSigmoidalMembershipFunctions");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ProductTwoSigmoidalMembershipFunctions");
```

```
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions1,0.0,10.0,0.1,CURVE_LINES,"
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions2,0.0,10.0,0.1,CURVE_LINES,"
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions3,0.0,10.0,0.1,CURVE_LINES,"
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (metodo Get)

Ottiene il primo rapporto di pendenza della funzione di appartenenza.

```
double  A1()
```

Valore di ritorno

Il primo rapporto di pendenza della funzione di appartenenza.

A1 (metodo Set)

Imposta il primo rapporto di pendenza della funzione di appartenenza.

```
void  A1(
   const double  a1      // il primo rapporto di pendenza della funzione di apparteneı
   )
```

Parametri

a1

[in] Il primo rapporto di pendenza della funzione di appartenenza.

A2 (metodo Get)

Ottiene il secondo rapporto di pendenza della funzione di appartenenza.

```
double  A2()
```

Valore di ritorno

Il secondo rapporto di pendenza della funzione di appartenenza.

A2 (metodo Set)

Imposta il secondo rapporto di pendenza della funzione di appartenenza.

```
void  A2(
   const double  a2      // il secondo rapporto di pendenza della funzione di apparter
   )
```

Parametri

a2

[in] Il secondo rapporto di pendenza della funzione di appartenenza.

C1 (metodo Get)

Ottiene il primo parametro della coordinata inflessione della funzione di appartenenza.

```
double  C1()
```

Valore di ritorno

La prima coordinata dell'inflessione della funzione di appartenenza.

C1 (metodo Set)

Imposta la prima coordinata dell'inflessione della funzione di appartenenza.

```
void  C1(
   const double  c1      // la prima coordinata dell' inflessione della funzione di ap
   )
```

Parametri

c1

[in] La prima coordinata dell' inflessione della funzione di appartenenza.

C2 (metodo Get)

Ottiene il secondo parametro della coordinata inflessione della funzione di appartenenza.

```
double  C2()
```

Valore di ritorno

La seconda coordinata dell'inflessione della funzione di appartenenza.

C2 (metodo Set)

Imposta il secondo parametro della coordinata di inflessione della funzione di appartenenza.

```
void  C2(
   const double  c2      // la seconda coordinata di inflessione della funzione di app
   )
```

Parametri

c2

[in] La seconda coordinata dell'inflessione della funzione funzione di appartenenza.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

Valore della funzione di appartenenza.

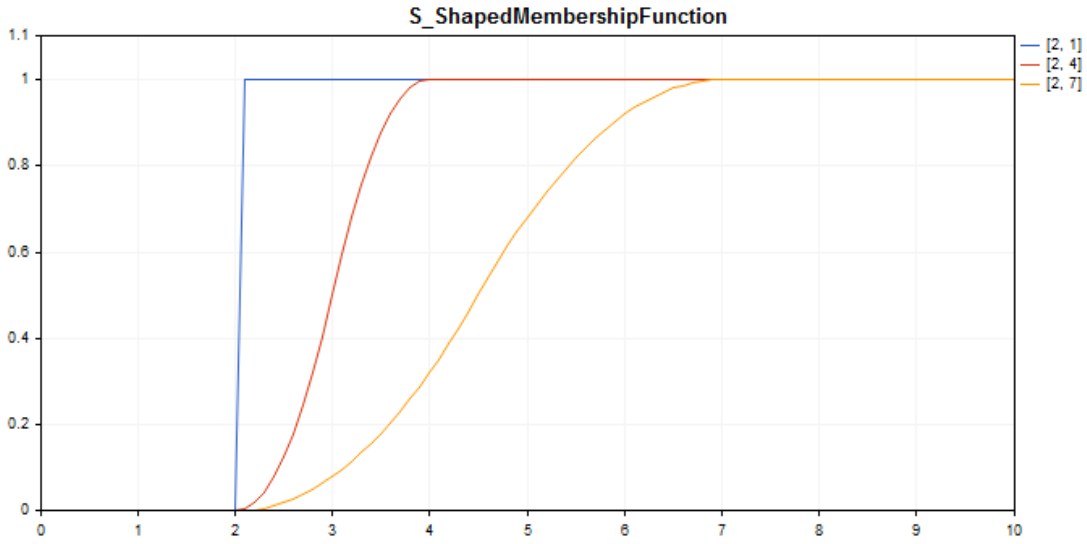
CS_ShapedMembershipFunction

Classe per implementare una funzione di appartenenza stile-S con i parametri A e B.

Descrizione

La funzione imposta una funzione di appartenenza a due parametri stile-S. Questa è una non decrescente funzione che assume valori da 0 ad 1. I parametri A e B definiscono l'intervallo all'interno del quale la funzione aumenta la traiettoria non lineare da 0 a 1.

La funzione rappresenta l'insieme fuzzy di tipo "molto alto" (cioè sono impostate funzioni di appartenenza non-decrescenti con saturazione).



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
  IMembershipFunction
    CS_ShapedMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
A	Ottiene ed imposta il parametro dell'inizio dell'intervallo crescente.

I metodi della classe	Descrizione
B	Ottiene ed imposta il primo parametro del nucleo dell' insieme fuzzy.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                    S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CS_ShapedMembershipFunction func1(2,1);
CS_ShapedMembershipFunction func2(2,4);
CS_ShapedMembershipFunction func3(2,7);
//--- Crea wrapper per funzioni di appartenenza
double S_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double S_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double S_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"S_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"S_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("S_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(S_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(S_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
   graphic.CurveAdd(S_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 7]");
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (metodo Get)

Ottiene il parametro dell'inizio dell'intervallo crescente

```
double  A()
```

Valore di ritorno

Il parametro dell'inizio dell'intervallo crescente

A (metodo Set)

Imposta il parametro dell'inizio dell'intervallo crescente.

```
void  A(
   const double  a      // parametro di inizio dell'intervallo crescente
   )
```

Parametri

a

[in] Parametro dell'inizio dell'intervallo crescente.

B (metodo Get)

Ottiene il primo parametro del core del set fuzzy.

```
double  B()
```

Valore di ritorno

Il primo parametro del nucleo del set fuzzy.

B (metodo Set)

Imposta il primo parametro del nucleo del set fuzzy.

```
void  B(
   const double  b      // il primo parametro del nucleo del set fuzzy
   )
```

Parametri

b

[in] Il primo parametro del nucleo del set fuzzy.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

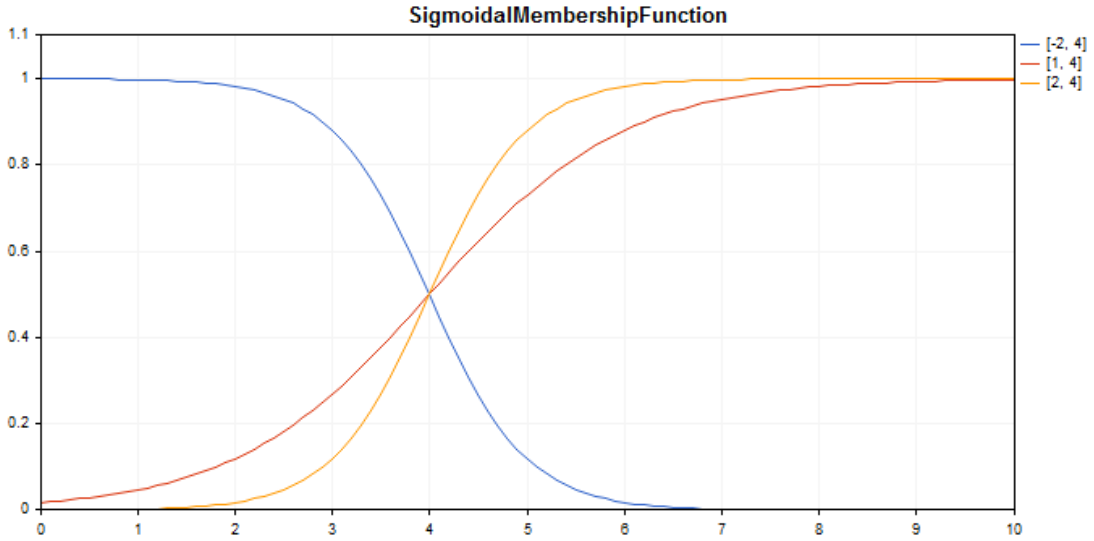
Valore della funzione di appartenenza.

CSigmoidalMembershipFunction

Classe per l'implementazione di una funzione di appartenenza sigmoide con i parametri A e C.

Descrizione

La funzione sigmoide viene applicata quando si impostano funzioni di appartenenza monotone. Essa consente la creazione di funzioni di appartenenza dei valori pari ad 1 che iniziano con un valore di argomento. Tali funzioni sono adatte se è necessario impostare tali termini linguistici come "short" o "long".



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject

IMembershipFunction

CSigmoidalMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
A	Ottiene ed imposta il rapporto pendenza funzione di appartenenza.

I metodi della classe	Descrizione
C	Ottiene ed imposta il parametro della coordinata inflessione della funzione di appartenenza.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                  SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CSigmoidalMembershipFunction func1(-2, 4);
CSigmoidalMembershipFunction func2(1, 4);
CSigmoidalMembershipFunction func3(2, 4);
//--- Crea wrapper per funzioni di appartenenza
double SigmoidalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double SigmoidalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double SigmoidalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"SigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"SigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("SigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(SigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[-2, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[1, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (metodo Get)

Ottiene il rapporto pendenza funzione di appartenenza.

```
double  A()
```

Valore di ritorno

Il rapporto pendenza della funzione di appartenenza.

A (metodo Set)

Imposta il rapporto pendenza funzione di appartenenza.

```
void  A(
   const double  a      // il primo rapporto di pendenza della funzione di appartenenz
   )
```

Parametri

a

[in] Il rapporto di pendenza della funzione di appartenenza.

C (metodo Get)

Ottiene il parametro coordinata dell'inflessione della funzione appartenenza.

```
double  C()
```

Valore di ritorno

Coordinare inflessione della funzione di appartenenza.

C (metodo Set)

Imposta la coordinata inflessione della funzione di appartenenza.

```
void  C(
   const double  c      // coordinata inflessione della funzione di appartenenza
   )
```

Parametri

c

[in] La coordinata inflessione della funzione di appartenenza.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

Valore della funzione di appartenenza.

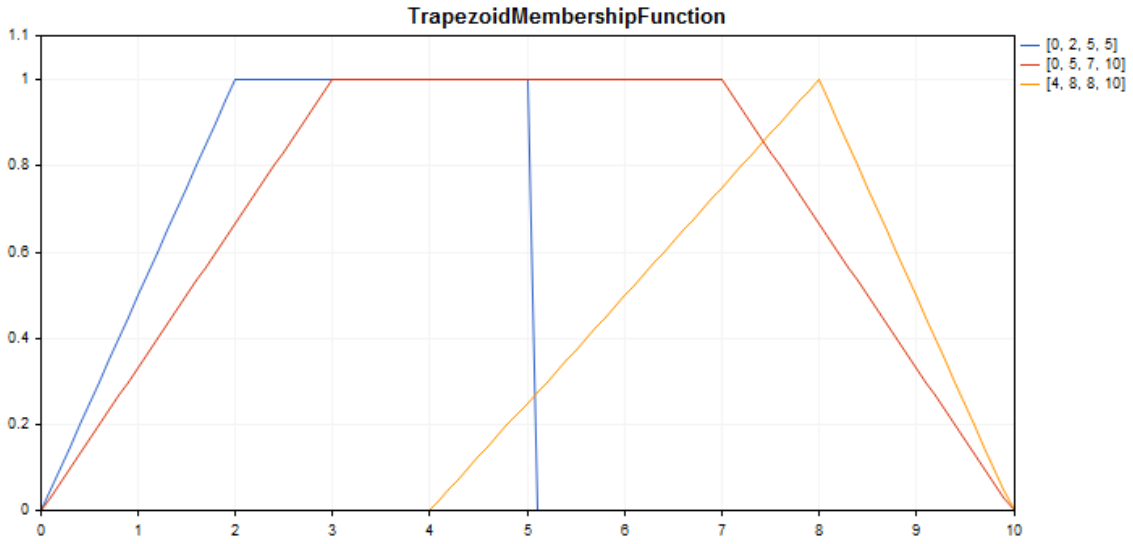
CTrapezoidMembershipFunction

Classe per implementare una funzione di appartenenza di forma trapezoidale con i parametri X1, X2, X3 e X4.

Descrizione

La funzione è formata utilizzando approssimazione lineare a tratti. Questa è una generalizzazione della funzione triangolare che consente di assegnare un nucleo di set fuzzy come intervallo. Tale funzione di appartenenza permette di interpretare comodamente stime ottimistiche/pessimistiche.

La funzione è utilizzata per impostare funzioni di appartenenza asimmetriche delle variabili con i valori più critici definiti entro un certo intervallo.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
  IMembershipFunction
    CTrapezoidMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
X1	Ottiene ed imposta il valore del primo punto sull'asse X.
X2	Ottiene ed imposta il valore del secondo punto sull'asse X.
X3	Ottiene ed imposta il valore del terzo punto sull'asse X.
X4	Ottiene ed imposta il valore del quarto punto sull'asse X.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                  TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CTrapezoidMembershipFunction func1(0,2,5,5);
CTrapezoidMembershipFunction func2(0,3,7,10);
CTrapezoidMembershipFunction func3(4,8,8,10);
//--- Crea wrapper per funzioni di appartenenza
double TrapezoidMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TrapezoidMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TrapezoidMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"TrapezoidMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TrapezoidMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TrapezoidMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
```

```
   graphic.CurveAdd(TrapezoidMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5, 5
   graphic.CurveAdd(TrapezoidMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 7, 1
   graphic.CurveAdd(TrapezoidMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[4, 8, 8, 1
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (metodo Get)

Ottiene il valore del primo punto sull'asse X.

```
double  X1()
```

Valore di ritorno

Il valore del primo punto sull'asse X.

X1 (metodo Set)

Imposta il valore del primo punto sull'asse X.

```
void  X1(
   const double  x1      // valore del primo punto sull'asse delle X
   )
```

Parametri

x1

[in] Il valore del primo punto sull'asse X.

X2 (metodo Get)

Ottiene il valore del secondo punto sull'asse X.

```
double  X2()
```

Valore di ritorno

Il valore del secondo punto sull'asse X.

X2 (metodo Set)

Imposta il valore del secondo punto sull'asse X.

```
void  X2(
   const double  x2      // valore del secondo punto sull'asse delle X
   )
```

Parametri

x2

[in] Il valore del secondo punto sull'asse delle X.

X3 (metodo Get)

Ottiene il valore del terzo punto sull'asse X.

```
double  X3()
```

Valore di ritorno

Il valore del terzo punto sull'asse X.

X3 (metodo Set)

Imposta il valore del terzo punto sull'asse X.

```
void  X3(
   const double  x3      // Valore del terzo punto sull'asse delle X
   )
```

Parametri

x3

[in] Il valore del terzo punto sull'asse delle X.

X4 (metodo Get)

Ottiene il valore del quarto punto sull'asse X.

```
double  X4()
```

Valore di ritorno

Il valore del quarto punto sull'asse X.

X4 (metodo Set)

Imposta il valore del quarto punto sull'asse X.

```
void  X4(
   const double  x4      // valore del quarto punto sull'asse delle X
   )
```

Parametri

x4

[in] Il valore del quarto punto sull'asse delle X.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

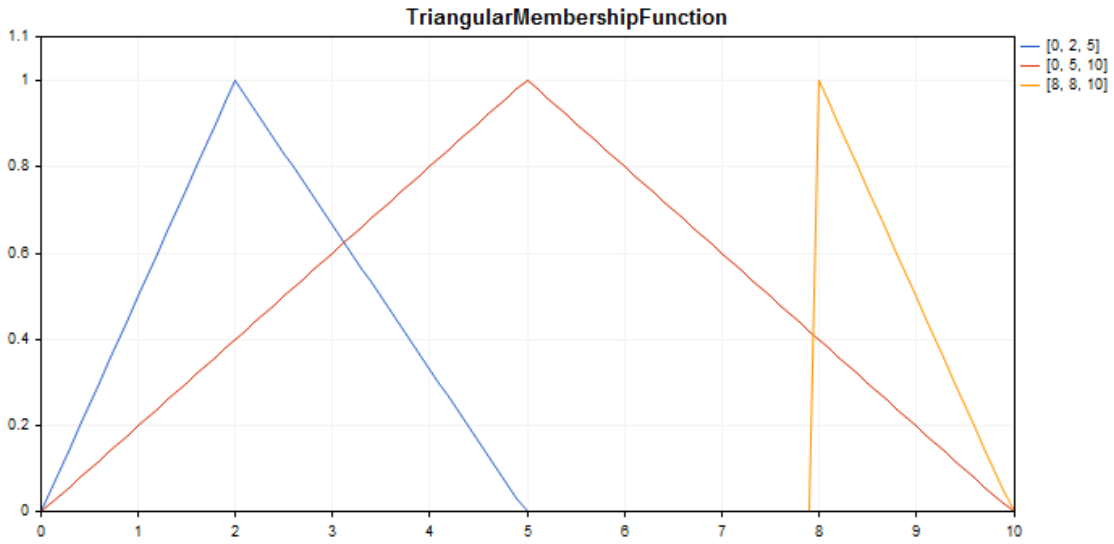
Valore della funzione di appartenenza.

CTriangularMembershipFunction

Classe per implementare una funzione di appartenenza triangolo con i parametri X1, X2 e X3.

Descrizione

La funzione imposta una funzione di appartenenza a forma di un triangolo. Si tratta di una funzione di appartenenza semplice e per la maggior parte applicata di frequente.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
IMembershipFunction
CTriangularMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
X1	Ottiene il valore del primo punto sull'asse X.
X2	Ottiene il valore del secondo punto sull'asse X.
X3	Ottiene il valore del terzo punto sull'asse X.

I metodi della classe	Descrizione
ToNormalMF	Converte una funzione di appartenenza triangolo in una Gaussiana.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                TriangularMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CTriangularMembershipFunction func1(0,2,5);
CTriangularMembershipFunction func2(0,5,10);
CTriangularMembershipFunction func3(8,8,10);
//--- Crea wrapper per funzioni di appartenenza
double TriangularMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TriangularMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TriangularMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"TriangularMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TriangularMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TriangularMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(TriangularMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5]"
   graphic.CurveAdd(TriangularMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 10]
   graphic.CurveAdd(TriangularMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[8, 8, 10]
```

```
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (metodo Get)

Ottiene il valore del primo punto sull'asse X.

```
double  X1()
```

Valore di ritorno

Il valore del primo punto sull'asse X.

X1 (metodo Set)

Imposta il valore del primo punto sull'asse X.

```
void  X1(
   const double  x1      // valore del primo punto sull'asse delle X
   )
```

Parametri

x1

[in] Il valore del primo punto sull'asse X.

X2 (metodo Get)

Ottiene il valore del secondo punto sull'asse X.

```
double  X2()
```

Valore di ritorno

Il valore del secondo punto sull'asse X.

X2 (metodo Set)

Imposta il valore del secondo punto sull'asse X.

```
void  X2(
```

```
   const double  x2      // valore del secondo punto sull'asse delle X
   )
```

Parametri

x2

[in] Il valore del secondo punto sull'asse delle X.

X3 (metodo Get)

Ottiene il valore del terzo punto sull'asse X.

```
double  X3()
```

Valore di ritorno

Il valore del terzo punto sull'asse X.

X3 (metodo Set)

Imposta il valore del terzo punto sull'asse X.

```
void  X3(
   const double  x3      // Valore del terzo punto sull'asse delle X
   )
```

Parametri

x3

[in] Il valore del terzo punto sull'asse delle X.

ToNormalMF

Converte una funzione di appartenenza triangolo in una Gaussiana.

```
CNormalMembershipFunction*  ToNormalMF()
```

Valore di ritorno

Il puntatore ad una funzione di appartenenza Gaussiana.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

Valore della funzione di appartenenza.

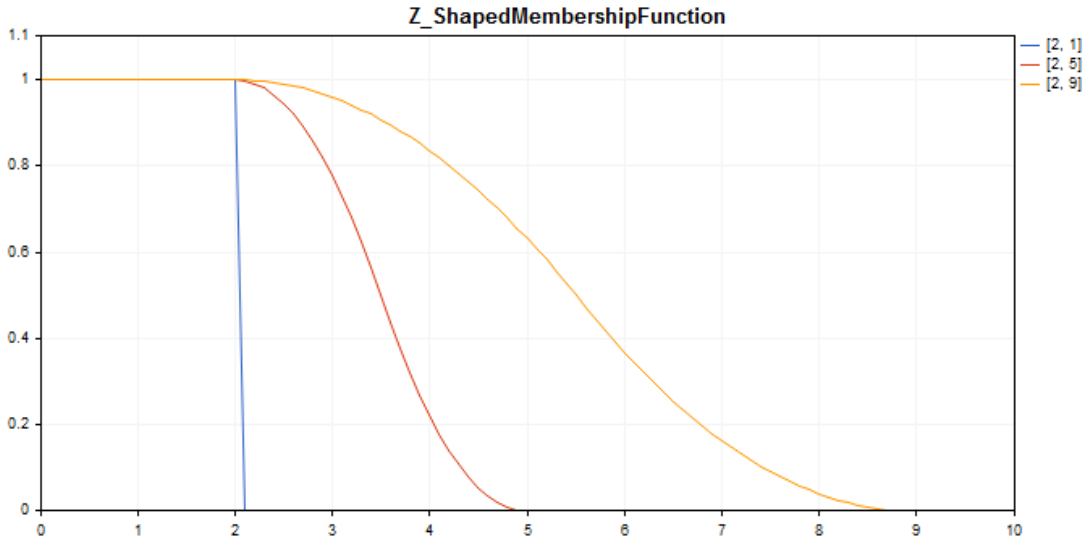
CZ_ShapedMembershipFunction

Classe per implementare una funzione di appartenenza stile-z ccon i parametri A e B.

Descrizione

La funzione imposta due parametri stile-z ad una funzione di appartenenza. Questa è una funzione di appartenenza non crescente che assume valori da 1 a 0. I parametri della funzione definiscono un intervallo entro il quale la funzione diminuisce di traiettoria non lineare da 1 a 0.

La funzione rappresenta insiemi fuzzy di tipo "molto basso". In altre parole, essa imposta funzioni di appartenenza non crescente con saturazione.



Un codice di esempio per tracciare un grafico viene visualizzato qui di seguito.

Dichiarazione

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject
IMembershipFunction
CZ_ShapedMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
A	Ottiene ed imposta il parametro della diminuzione inizio dell'intervallo.

I metodi della classe	Descrizione
B	Ottiene ed imposta il parametro della diminuzione fine dell'intervallo.
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Esempio

```
//+------------------------------------------------------------------+
//|                                  Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Crea funzioni di appartenenza
CZ_ShapedMembershipFunction func1(2,1);
CZ_ShapedMembershipFunction func2(2,5);
CZ_ShapedMembershipFunction func3(2,9);
//--- Crea wrapper per funzioni di appartenenza
double Z_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double Z_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double Z_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Funzione di start del programma Script                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"Z_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"Z_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("Z_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- crea curva
   graphic.CurveAdd(Z_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(Z_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 5]");
   graphic.CurveAdd(Z_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 9]");
//--- imposta le proprietà dell'asse X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- imposta le proprietà asse Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plotta
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (metodo Get)

Ottiene il parametro della diminuzione inizio dell'intervallo.

```
double  A()
```

Valore di ritorno

Parametro decrescente di avvio intervallo.

A (metodo Set)

Imposta il parametro decrescente di inizio dell'intervallo.

```
void  A(
   const double  a      // parametro decrescente inizio intervallo
   )
```

Parametri

a

[in] Parametro decrescente inizio intervallo.

B (metodo Get)

Ottiene il parametro della diminuzione fine dell'intervallo.

```
double  B()
```

Valore di ritorno

Parametro diminuendo di fine dell'intervallo.

B (metodo Set)

Imposta il parametro decrescente di fine dell'intervallo.

```
void  B(
   const double  b      // parametro decrescente fine dell'intervallo
   )
```

Parametri

b

[in] Parametro decrescente fine dell'intervallo.

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

Valore della funzione di appartenenza.

IMembershipFunction

Classe di base per tutte le classi di funzione di appartenenza.

Dichiarazione

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Titolo

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Gerarchia di ereditarietà

CObject

IMembershipFunction

Discendenti diretti

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

I metodi della classe

I metodi della classe	Descrizione
GetValue	Calcola il valore della funzione di appartenenza da un argomento specificato.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

Calcola il valore della funzione di appartenenza da un argomento specificato.

```
double  GetValue(
   const  x      // argomento della funzione di appartenenza
   )
```

Parametri

x

[in] Argomento funzione di appartenenza.

Valore di ritorno

Valore della funzione di appartenenza.

Regole sistemi fuzzy

Un **sistema fuzzy** (sistema logico a inferenza fuzzy) è una ricevuta di conclusione nella forma di un insieme fuzzy corrispondenti ai valori correnti degli input con l'uso di una serie di **regole fuzzy** ed operazioni fuzzy.

Le **regole fuzzy** determinano la relazione tra input ed output di un oggetto in esame. La quantità di regole nel sistema è illimitato. Il formato generalizzato di regole fuzzy è il seguente:

se*condizione della regola*, **allora** *conclusione della regola*.

Condizione della regola descrive lo stato corrente dell'oggetto. *Conclusione della regola* descrive come la condizione ha effetto sull'oggetto.

Classe di regole per sistemi fuzzy	Descrizione
CMamdaniFuzzyRule	Classe per l'attuazione di una regola logica fuzzy per l'algoritmo Mamdani
CSugenoFuzzyRule	Classe per l'attuazione di una regola logica fuzzy per l'algoritmo Sugeno
CSingleCondition	La classe imposta una condizione fuzzy espressa dalla coppia "Variabile Fuzzy — Termine Fuzzy".
CConditions	LA Classe definisce un insieme di condizioni Fuzzy collegate tra loro da un operatore.
CGenericFuzzyRule	Classe base per l'attuazione dei due tipi di regole fuzzy.

CMamdaniFuzzyRule

Inferenza fuzzy di tipo-Mamdani — uno dei due tipi fondamentali di sistemi fuzzy. I valori delle variabili di output sono impostati utilizzando termini fuzzy.

Descrizione

La regola logica Fuzzy per l'algoritmo Mamdani può essere descritta come segue:

$$if(X_1 \ is \ a_1) \wedge (X_2 \ is \ a_2) \wedge \ldots \wedge (X_n \ is \ a_n) \ then \ (Y \ is \ d)(W)$$

dove:

- X = (X1, X2, X3 ... Xn) — vettore di variabili di input;
- Y — variabile output;
- a = (a1, a2, a3 ... an) — vettore di valori variabili input;
- d — valore variabile output;
- W — peso della regola.

Dichiarazione

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

Titolo

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Gerarchia di ereditarietà

CObject
IParsableRule
CGenericFuzzyRule
CMamdaniFuzzyRule

I metodi della classe

I metodi della classe	Descrizione
Conclusion	Ottiene ed imposta la conclusione della regola fuzzy Mamdani
Weight	Ottiene e imposta il peso della regola fuzzy Mamdani

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (metodo Get)

Ottiene la conclusione della regola fuzzy Mamdani

```
CSingleConditon*  Conclusion()
```

Valore di ritorno

Conclusione della regola fuzzy Mamdani.

Conclusion (metodo Set)

Imposta la conclusione della regola fuzzy Mamdani.

```
void  Conclusion(
   CSingleConditon*  value      // conclusione della regola fuzzy Mamdani
   )
```

Parametri

valore

[in] Conclusione della regola fuzzy Mamdani.

Weight (metodo Get)

Ottiene il peso della regola fuzzy Mamdani.

```
double  Weight()
```

Valore di ritorno

Peso della regola fuzzy Mamdani.

Weight (metodo Set)

Imposta il peso della regola fuzzy Mamdani.

```
void  Weight(
   const double  value      // peso della regola fuzzy Mamdani
   )
```

Parametri

valore

[in] Peso della regola fuzzy Mamdani.

CSugenoFuzzyRule

Inferenza fuzzy di tipo Sugeno — uno dei due tipi fondamentali di sistemi fuzzy. i valori delle variabili di output sono impostati come combinazione lineare di variabili di input.

Descrizione

A differenza della regola Mamdani, un valore variabile di input è impostato da una funzione lineare da voci anziché da un termine fuzzy. La regola logica Fuzzy per l'algoritmo Sugeno può essere descritta come segue:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

dove:

- X = (X1, X2, X3 ... Xn) — vettore di variabili di input;
- Y — variabile output;
- a = (a1, a2, a3 ... an) — vettore di valori variabili input;
- b = (b1, b2, b3 ... bn) — rapporto termine libero nella funzione lineare per un valore di output
- W — peso della regola.

Dichiarazione

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

Titolo

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Gerarchia di ereditarietà

CObject
IParsableRule
CGenericFuzzyRule
CSugenoFuzzyRule

I metodi della classe

I metodi della classe	Descrizione
Conclusion	Ottiene ed imposta la conclusione della regola fuzzy Sugeno

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (metodo Get)

Ottiene la conclusione della regola fuzzy Sugeno.

```
CSingleConditon*  Conclusion()
```

Valore di ritorno

Conclusione della regola fuzzy Sugeno.

Conclusion (metodo Set)

Imposta la conclusione della regola fuzzy Sugeno.

```
void  Conclusion(
   CSingleCondition*  value      // conclusione della regola fuzzy Sugeno
   )
```

Parametri

valore

[in] Conclusione della regola fuzzy Sugeno.

CSingleCondition

La classe imposta una condizione fuzzy espressa dalla coppia "Variabile Fuzzy — Termine Fuzzy".

Descrizione

Secondo una condizione fuzzy, una variabile corrisponde ad un termine. Una condizione fuzzy può essere descritta dalla seguente espressione: *X è a*,

dove:

- X è una variabile fuzzy;
- a è una valore della variabile fuzzy (termine fuzzy).

Dichiarazione

```
class CSingleCondition : public ICondition
```

Titolo

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Gerarchia di ereditarietà

CObject

ICondition

CSingleCondition

Discendenti diretti

CFuzzyCondition

I metodi della classe

I metodi della classe	Descrizione
Not	Ottiene ed imposta il flag che indica se è necessario applicare negazione a questa condizione.
Term	Ottiene ed imposta un termine fuzzy per questa condizione.
Var	Ottiene ed imposta una variabile fuzzy per questa condizione.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not (metodo Get)

Ottiene il flag che indica se è necessario applicare negazione a questa condizione.

```
bool Not()
```

Valore di ritorno

Il valore del flag.

Not (metodo Set)

Imposta il flag che indica se è necessario applicare negazione a questa condizione.

```
void  Not(
   bool  not      // valore flag
   )
```

Parametri

not

[in] Valore Flag.

Term (metodo Get)

Ottiene un termine fuzzy per la condizione data.

```
INamedValue* Term()
```

Valore di ritorno

Un termine fuzzy per la condizione data.

Term (metodo Set)

Imposta un termine fuzzy per la condizione data.

```
void  Term(
   INamedValue*&  value      // termine fuzzy per la condizione data
   )
```

Parametri

valore

[in] Un termine fuzzy per la condizione data.

Var (metodo Get)

Ottiene una variabile fuzzy per la condizione data.

```
INamedVariable* Var()
```

Valore di ritorno

Una variabile fuzzy per la condizione data.

Var (metodo Set)

Imposta una variabile fuzzy la condizione data

```
void  Var(
   INamedVariable*&  value      // una variabiile fuzzy per la condizione data
   )
```

Parametri

valore

[in] variabile fuzzy.

CConditions

LA Classe definisce un insieme di condizioni Fuzzy collegate tra loro da un operatore.

Descrizione

Un insieme di condizioni Fuzzy collegate tra loro da un operatore può essere descritto come segue:

$$(X_1 \; is \; a_1) \wedge (X_2 \; is \; a_2) \wedge \ldots \wedge (X_n \; is \; a_n)$$

dove:

- X = (X1, X2, X3 ... Xn) — vettore di variabili di input;
- a = (a1, a2, a3 ... an) — vettore dei valori delle variabili di input.

In questo esempio, viene utilizzato l'operatore *and*. Inoltre, l'operatore *or* è disponibile in questa classe.

Dichiarazione

```
class CConditions : public ICondition
```

Titolo

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Gerarchia di ereditarietà

CObject
  ICondition
    CConditions

I metodi della classe

I metodi della classe	Descrizione
ConditionsList	Ottiene l'elenco di tutte le condizioni
Not	Ottiene ed Imposta il flag che indica se è necessario applicare negazione a queste condizioni
Op	Gets and sets a type of the conditions bundle operator.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

Ottiene l'elenco di tutte le condizioni.

```
CList*  ConditionsList()
```

Valore di ritorno

Elenco di tutte le condizioni.

Not (metodo Get)

Ottiene il flag che indica se è necessario applicare negazione a queste condizioni.

```
bool Not()
```

Valore di ritorno

Il valore del flag.

Not (metodo Set)

Imposta il flag che indica se è necessario applicare negazione a queste condizioni

```
void  Not(
   bool  not      // valore flag
   )
```

Parametri

not

[in] Valore Flag.

Op (metodo Get)

Gets a type of the conditions bundle operator. Gli operatori *and* ed *or* sono disponibili.

```
OperatorType Op()
```

Valore di ritorno

Tipo di operatore condizionale.

Op (metodo Set)

Imposta la condizionale operatore. Gli operatori *and* ed *or* sono disponibili.

```
void  Op(
   OperatorType  op     // tipo di operatore condizionale
   )
```

Parametri

op

[in] Tipo di operatore condizionale.

CGenericFuzzyRule

Classe base per entrambi i tipi di regole fuzzy.

Dichiarazione

```
class CGenericFuzzyRule : public IParsableRule
```

Titolo

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Gerarchia di ereditarietà

CObject

IParsableRule

CGenericFuzzyRule

Discendenti diretti

CMamdaniFuzzyRule, CSugenoFuzzyRule

I metodi della classe

I metodi della classe	Descrizione
Conclusion	Ottiene ed imposta la regola conclusione Fuzzy
Condition	Ottiene ed imposta la condizione (if) 'se' (insieme di condizioni) per una regola fuzzy
CreateCondition	Crea una condizione per una regola fuzzy da parametri specificati

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion (metodo Get)

Ottiene la conclusione di una regola fuzzy.

```
CSingleConditon*  Conclusion()
```

Valore di ritorno

Conclusione di un regola fuzzy.

Conclusion (metodo Set)

Imposta la conclusione della regola fuzzy.

```
virtual void  Conclusion(
   CSingleConditon*  value      // conclusione della regola fuzzy
```

```
   )
```

Parametri

valore

[in] Conclusione di una regola fuzzy.

Condition (Get method)

Ottiene la condizione (if) 'se' (insieme di condizioni) per una regola fuzzy.

```
CConditons*  Condition()
```

Valore di ritorno

Condizione Fuzzy (insieme di condizioni).

Condition (metodo Set)

Imposta la condizione (if) 'se' (insieme di condizioni) per una regola fuzzy.

```
void  Condition(
   CConditons*  value      // condizione 'if' (set di condizioni) per una regola fuzzy
   )
```

Parametri

valore

[in] Condizione Fuzzy (set di condizioni).

CreateCondition

Crea una condizione per una regola fuzzy dai parametri specificati.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // variabile fuzzy
   CFuzzyTerm*      term,      // termine fuzzy
   )
```

Parametri

var

[in] Variabile Fuzzy.

term

[in] Termine Fuzzy.

Valore di ritorno

Stato della regola Fuzzy.

CreateCondition

Crea una condizione per una regola fuzzy dai parametri specificati.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // variabile fuzzy
   CFuzzyTerm*      term,     // termine fuzzy
   bool             not,      // flag indicante se è necessario applicare negazione ac
   )
```

Parametri

var

[in] Variabile Fuzzy.

term

[in] Termine Fuzzy.

not

[in] Flag che indica se è necessario applicare la negazione ad una condizione.

Valore di ritorno

Stato della regola Fuzzy.

CreateCondition

Crea una condizione per una regola fuzzy dai parametri specificati.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,      // variabile fuzzy
   CFuzzyTerm*     term,     // termine fuzzy
   bool            not,      // flag che indica se è necessario applicare negazione ac
   HedgeType       hedge     // tipo bundle condizione
   )
```

Parametri

var

[in] Variabile Fuzzy.

term

[in] Termine Fuzzy.

not

[in] Flag che indica se è necessario applicare la negazione ad una condizione.

hedge

[in] Tipo di bundle condizione.

Valore di ritorno

Stato della regola Fuzzy.

Variabili sistemi fuzzy

Fuzzy (linguistiche) variabili **sono applicati a sistemi fuzzy**. Queste sono le variabili i cui valori sono parole o combinazioni di parole in un linguaggio naturale o artificiale.

Variabili linguistiche comprendono insiemi fuzzy. La natura e il numero di variabili fuzzy cambiano per ogni determinato compito nel definire insiemi fuzzy.

Classe	Descrizione
CFuzzyVariable	Classe per la creazione di variabili fuzzy generali.
CSugenoVariable	Classe per la creazione di variabili fuzzi di tipo Sugeno.

CFuzzyVariable

Classe per la creazione di variabili fuzzy generali.

Descrizione

Qui, una variabile fuzzy viene creata con i seguenti parametri:

- valore massimo della variabile;
- valore minimo della variabile;
- nome variabile fuzzy ;
- term set (insieme di tutti i possibili valori che una variabile linguistica è in grado di ricevere).

Dichiarazione

```
class CFuzzyVariable : public CNamedVariableImpl
```

Titolo

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

Gerarchia di ereditarietà

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CFuzzyVariable

I metodi della classe

I metodi della classe	Descrizione
AddTerm	Aggiunge un singolo termine fuzzy per una variabile fuzzy.
GetTermByName	Ottiene un termine fuzzy da un nome specificato.
Max	Ottiene ed imposta un valore massimo per una variabile fuzzy.
Min	Ottiene ed imposta un valore minimo per una variabile fuzzy.
Terms	Ottiene ed imposta un elenco di termini fuzzy per la variabile fuzzy data.
Valori	Ottiene ed imposta un elenco di termini fuzzy per la variabile fuzzy data.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CNamedVariableImpl

Name, Name

AddTerm

Aggiunge un singolo termine fuzzy per una variabile fuzzy.

```
void  AddTerm(
   CFuzzyTerm*&  term      // termine fuzzy
   )
```

Parametri

term

[in] Termine Fuzzy.

GetTermByName

Ottiene un termine fuzzy da un nome specificato.

```
CFuzzyTerm*  GetTermByName(
   const string  name      // nome termine fuzzy
   )
```

Parametri

name

[in] Nome termine Fuzzy.

Valore di ritorno

Termine Fuzzy con un nome specificato.

Max (metoto Get)

Ottiene il valore massimo per una variabile fuzzy.

```
double  Max()
```

Valore di ritorno

Valore massimo per una variabile fuzzy.

Max (metodo Set)

Imposta il valore massimo per una variabile fuzzy.

```
void  Max(
   const double  max      // massimo valore per una variabile fuzzy
```

```
   )
```

Parametri

max

[in] Massimo valore per una variabile fuzzy.

Min (metodo Get)

Ottiene il valore minimo per una variabile fuzzy.

```
double  Min()
```

Valore di ritorno

Valore minimo per una variabile fuzzy.

Max (metodo Set)

Imposta il valore minimo per una variabile fuzzy.

```
void  Min(
   const double  min      // valore minimo per una variabile fuzzy
   )
```

Parametri

min

[in] Valore minimo per una variabile fuzzy.

Terms (metodod Get)

Ottiene un elenco di termini fuzzy per la variabile fuzzy data.

```
CList*  Terms()
```

Valore di ritorno

Elenco dei termini fuzzy per la variabile fuzzy data.

Terms (metodo Set)

Imposta un elenco di termini fuzzy per la variabile fuzzy data.

```
void  Terms(
   CList*&  terms      // lista di termini fuzzy per la variabile fornita
   )
```

Parametri

terms

[in] Elenco di termini fuzzy per la variabile fuzzy fornita.

Valori

Ottiene un elenco di termini fuzzy per la variabile fuzzy data.

```
CList*  Values()
```

Valore di ritorno

Elenco dei termini fuzzy per la variabile fornita.

CSugenoVariable

Classe per la creazione di variabili fuzzi di tipo Sugeno.

Descrizione

La variabile fuzzy di tipo-Sugeno è diversa dalla variabile linguistica generale poiché non viene impostata da un set di termini ma da un insieme di funzioni lineari.

Dichiarazione

```
class CSugenoVariable : public CNamedVariableImpl
```

Titolo

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

Gerarchia di ereditarietà

CObject
INamedValue
INamedVariable
CNamedVariableImpl
CSugenoVariable

I metodi della classe

I metodi della classe	Descrizione
Functions	Ottiene l'elenco delle funzioni lineari della variabile fuzzy Sugeno.
GetFuncByName	Ottiene la funzione lineare con un nome specificato.
Valori	Ottiene l'elenco delle funzioni lineari della variabile fuzzy Sugeno.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CNamedVariableImpl

Name, Name

Functions

Ottiene l'elenco delle funzioni lineari della variabile fuzzy Sugeno.

```
CList*  Functions()
```

Valore di ritorno

Elenco delle funzioni lineari.

GetFuncByName

Ottiene la funzione lineare con un nome specificato.

```
ISugenoFunction*  GetFuncByName(
   const string  name      // Nome funzione lineare
   )
```

Parametri

name

[in] Nome funzione lineare.

Valore di ritorno

Funzione lineare con un nome specificato.

Valori

Ottiene l'elenco delle funzioni lineari della variabile fuzzy Sugeno.

```
CList*  Values()
```

Valore di ritorno

Elenco delle funzioni lineari della variabile fuzzy Sugeno.

CFuzzyTerm (fuzzy terms)

Classe per l'attuazione termini fuzzy.

Descrizione

Un termine è qualsiasi elemento di un set di termini. Un termine è definito da due componenti:

- nome fuzzyterm;
- funzione di appartenenza.

Dichiarazione

```
class CFuzzyTerm : public CNamedValueImpl
```

Titolo

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

Gerarchia di ereditarietà

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

I metodi della classe

I metodi della classe	Descrizione
MembershipFunction	Ottiene una funzione di appartenenza per il fuzzyterm.

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CNamedValueImpl

Name, Name

MembershipFunction

Ottiene una funzione di appartenenza per il fuzzyterm.

```
IMembershipFunction*  MembershipFunction()
```

Valore di ritorno

Funzione di appartenenza

Sistemi Fuzzy

Il sistema Fuzzy (o *modello fuzzy*) e' un modello matematico il cui calcolo è basato su logica fuzzy. La costruzione di tali modelli è applicabile quando l'oggetto di studio ha una formalizzazione debole e la sua esatto descrizione matematica è troppo complessa o sconosciuta.

Il progresso di una costruzione del modello può essere suddiviso in tre fasi principali:

1. Definizione delle caratteristiche di input ed output di un modello.
2. Costruire una base di conoscenze.
3. La selezione di uno dei metodi di inferenza Fuzzy (Mamdani e Sugeno).

La prima fase effettua direttamente le conseguenti due e determina il futuro funzionamento del modello.

Una conoscenza di base (*base di regole*) e' un insieme di regole fuzzy di tipo "se, allora" che definiscono il rapporto tra input ed output dell'oggetto esaminato.

Condizione della regola descrive lo stato corrente dell'oggetto, e **la conclusione della regola** — come questa condizione affligge l'oggetto.

Ci possono essere due tipi di termini e conclusioni per ogni regola:

1. semplice (link a Csinglcond) — include una variabile fuzzy;
2. complesso (link Cconditions) — include diverse variabili fuzzy.

Ogni regola nel sistema ha il suo **peso(weight)** — importanza di una regola nel modello. Fattori di ponderazione(pesatura) sono assegnati ad una regola nel range [0, 1].

A seconda della base di conoscenza creata, il sistema di inferenza fuzzy viene determinato per un modello. **La inferenza logica Fuzzy** è una ricevuta di conclusione in forma di un insieme fuzzy corrispondente al valore attuale degli inputs con l' uso di conoscenza base ed operazioni fuzzy. I due tipi principali di inferenza Fuzzy sono Mamdani e Sugeno.

Mamdani system

I valori delle variabili di output del sistema Mamdani vengono impostati mediante termini fuzzy.

Descrizione

La regola logica Fuzzy per l'algoritmo Mamdani può essere descritta come segue:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

dove:

- X = (X1, X2, X3 ... Xn) — vettore di variabili di input;
- Y — variabile output;
- a = (a1, a2, a3 ... an) — vettore di valori variabili input;
- d — valore variabile output;
- W — peso della regola.

I metodi della classe

I metodi della classe	Descrizione
AggregationMethod	Imposta il tipo di condizioni di aggregazione
Calculate	Calcola un' inferenza fuzzy per il sistema
DefuzzificationMethod	Imposta tipo di metodo di defuzzificazione
EmptyRule	Crea una regola Mamdani Fuzzy vuota basato sul sistema attuale
ImplicationMethod	Imposta un tipo di sistema implicazione operatore
Output	Ottiene l'elenco delle variabili output fuzzy Mamdani.
OutputByName	Ottiene una variabile output fuzzy Mamdani da un nome specificato.
ParseRule	Crea una regola Mamdani fuzzy sulla base di una linea specificata.
Rules	Restituisce l'elenco delle regole fuzzy Mamdani.

Metodi ereditati dalla classe CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

Imposta il tipo di metodo di condizioni di aggregazione.

```
void  AggregationMethod(
   AggregationMethod  value      // tipo di metodo di aggregazopne
```

```
   )
```

Parametri

valore

[in] Tipo di metodo condizioni di aggregazione.

Calculate

Calcola un' inferenza fuzzy per il sistema

```
CList*  Calculate(
   CList*  inputValues      // dati input
   )
```

Parametri

inputValues

[in] Dati input per il calcolo.

Valore di ritorno

Risultato del calcolo.

DefuzzificationMethod

Imposta tipo di metodo di defuzzificazione.

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // tipo di metodo di defuzzificazione.
   )
```

Parametri

valore

[in] Tipo di metodo di defuzzificazione.

EmptyRule

Crea una regola Mamdani Fuzzy vuota basato sul sistema attuale

```
CMamdaniFuzzyRule*  EmptyRule()
```

Valore di ritorno

Regola Mamdani Fuzzy.

ImplicationMethod

Imposta un tipo di condizione implicazione operatore.

```
void  ImplicationMethod(
   ImplicationMethod  value      // tipo di condizione implicazione operatore
```

```
   )
```

Parametri

valore

[in] Tipo di condizioni implicazione operatore.

Output

Ottiene l'elenco delle variabili output fuzzy Mamdani.

```
CList*  Output()
```

Valore di ritorno

Elenco delle variabili fuzzy.

OutputByName

Ottiene una variabile output fuzzy Mamdani da un nome specificato.

```
CFuzzyVariable*  OutputByName(
   const string  name      // nume variabile fuzzy
   )
```

Parametri

name

[in] Nome variabile Fuzzy.

Valore di ritorno

Variabile Fuzzy Mamdani con il nome specificato.

ParseRule

Crea una regola Mamdani fuzzy sulla base di una linea specificata.

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // rappresentazione stringa di una regola fuzzy
   )
```

Parametri

rule

[in] Rappresentazione stringa di una regola fuzzy Mamdani.

Valore di ritorno

Regola Mamdani Fuzzy.

Rules

Restituisce l'elenco delle regole fuzzy Mamdani.

```
CList*  Rules()
```

Valore di ritorno

Elenco delle regole fuzzy Mamdani.

CSugenoFuzzyRule

Il sistema logico fuzzy Sugeno è uno dei due tipi fondamentali di sistemi fuzzy. i valori delle variabili di output sono impostati come combinazione lineare di variabili di input.

Descrizione

A differenza della regola Mamdani, un valore variabile di input è impostato da una funzione lineare da voci anziché da un termine fuzzy. La regola logica Fuzzy per l'algoritmo Sugeno può essere descritta come segue:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

dove:

- X = (X1, X2, X3 ... Xn) — vettore di variabili di input;
- Y — variabile output;
- a = (a1, a2, a3 ... an) — vettore di valori variabili input;
- b = (b1, b2, b3 ... bn) — rapporto termine libero nella funzione lineare per un valore di output
- W — peso della regola.

I metodi della classe

I metodi della classe	Descrizione
Calculate	Calcola un' inferenza fuzzy per il sistema
CreateSugenoFunction	Crea una funzione Sugeno lineare per il sistema.
EmptyRule	Crea una regola Sugeno Fuzzy vuota basata sul sistema attuale.
Output	Ottiene l'elenco delle variabili di output fuzzy Sugeno.
OutputByName	Ottiene una variabile di uscita fuzzy Sugeno da un nome specificato.
ParseRule	Crea una regola fuzzy Sugeno sulla base di una linea specificata.
Rules	Restituisce l'elenco delle regole fuzzy.

Metodi ereditati dalla classe CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

Calcola un' inferenza fuzzy per il sistema

```
CList*  Calculate(
   CList*&  inputValues      // dati input
   )
```

Parametri

inputValues

[in] Dati input per il calcolo.

Valore di ritorno

Risultato del calcolo.

CreateSugenoFunction

Crea una funzione Sugeno lineare per il sistema.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,           // nome della funzione
   const double& coeffs[]        // rapporti della funzione
   )
```

Parametri

name

[in] Nome della Funzione.

coeffs[]

[in] Rapporti della Funzione.

Valore di ritorno

Funzione lineare Sugeno.

Nota

La grandezza dell'array rapporto può essere uguala ad un numero di input o superiore a tale numero di un' unità. Nel primo caso, il termine libero della funzione lineare Sugeno è uguale a zero, mentre nel secondo caso è uguale all'ultimo rapporto.

CreateSugenoFunction

Crea una funzione Sugeno lineare per il sistema.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,           // nome della funzione
   CList*&       coeffs,         // lista di coppie variabili fuzzy - il rapporto
   const double  constValue      // rapporto del termine libero della funzione
   )
```

Parametri

name

[in] Nome della Funzione.

coeffs[]

[in] Rapporti della Funzione.

Valore di ritorno

Funzione lineare Sugeno.

EmptyRule

Crea una regola Sugeno Fuzzy vuota basata sul sistema attuale.

```
CSugenoFuzzyRule*  EmptyRule()
```

Valore di ritorno

Regola Sugeno Fuzzy.

Output

Ottiene l'elenco delle variabili di output fuzzy Sugeno.

```
CList*  Output()
```

Valore di ritorno

Elenco delle variabili fuzzy.

OutputByName

Ottiene una variabile di uscita fuzzy Sugeno da un nome specificato.

```
CSugenoVariable*  OutputByName(
   const string  name      // nume variabile fuzzy
   )
```

Parametri

name

Nome della variabile Fuzzy.

Valore di ritorno

Variabile Fuzzy Sugeno con un nome specificato.

ParseRule

Crea una regola fuzzy Sugeno sulla base di una linea specificata.

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // rappresentazione stringa di una regola Fuzzy Sugeno
   )
```

Parametri

rule

[in] Rappresentazione Stringa di una regola fuzzy Sugeno.

Valore di ritorno

Regola Sugeno Fuzzy.

Rules

Restituisce l'elenco delle regole fuzzy.

```
CList*  Rules()
```

Valore di ritorno

Elenco di regole fuzzy.

Classe per lavorare con i programmi OpenCL

La classe COpenCL è un wrapper per facilitare il lavoro con le funzioni OpenCL. In alcuni casi, l'uso della GPU permette di aumentare notevolmente la velocità di calcolo.

Esempi di utilizzo della classe per i calcoli basati su valori float e double possono essere trovati nelle corrispondenti sottodirectory dei MQL5\Scripts\Examples\OpenCL\cartella\. I codici sorgente dei programmi OpenCL si trovano nelle sottodirectory MQL5\Scripts\Examples\OpenCL\Double\Kernels e MQL5\Scripts\Examples\OpenCL\Float\Kernels.

- MatrixMult.mq5 - esempio di moltiplicazione di matrici utilizzando la memoria globale e locale
- BitonicSort.mq5 - esempio di ordinamento parallelo di elementi dell'array nella GPU
- FFT.mq5 - esempio di calcolo di trasformata rapida di Fourier
- Wavelet.mq5 - esempio di trasformata wavelet dei dati utilizzando la Wavelet Morlet.

Si raccomanda di scrivere il codice sorgente per OpenCL in file separati CL, che possono essere successivamente inseriti nel programma MQL5 utilizzando le variabili di risorse.

Dichiarazione

```
class COpenCL
```

Titolo

```
#include <OpenCL\OpenCL.mqh>
```

Metodi della classe

Nome	Descrizione
BufferCreate	Crea un buffer OpenCL in corrispondenza dell'indice specificato
BufferFree	Elimina buffer in corrispondenza dell'indice specificato
BufferFromArray	Crea un buffer in corrispondenza dell'indice specificato da un array di valori
BufferRead	Legge un buffer OpenCL all'indice specificato in un array
BufferWrite	Scrive un array di valori nel buffer in corrispondenza dell'indice specificato
Execute	Esegue il kernel OpenCL con l'indice specificato
GetContext	Restituisce l' handle del contesto OpenCL
GetKernel	Restituisce l'handle dell'oggetto kernel in corrispondenza dell'indice specificato
GetKernelName	Restituisce il nome dell'oggetto kernel in corrispondenza dell'indice specificato
GetProgram	Restituisce l'handle del programma OpenCL
Initialize	Inizializza il programma OpenCL

Nome	Descrizione
KernelCreate	Crea un punto di ingresso nel programma OpenCL in corrispondenza dell'indice specificato
KernelFree	Rimuove una funzione di avvio OpenCL in corrispondenza dell'indice specificato
SetArgument	Imposta un parametro per la funzione OpenCL in corrispondenza dell'indice specificato
SetArgumentBuffer	Imposta un buffer OpenCL come parametro della funzione OpenCL in corrispondenza dell'indice specificato
SetArgumentLocalMemory	Imposta un parametro nella memoria locale per la funzione di OpenCL in corrispondenza dell'indice specificato
SetBuffersCount	Imposta il numero di buffer
SetKernelsCount	Imposta il numero di oggetti del kernel
Shutdown	Scarica il programma OpenCL
SupportDouble	Controlla se i tipi di dati in virgola mobile sono supportati dal dispositivo

BufferCreate

Crea un buffer OpenCL in corrispondenza dell'indice specificato.

```
bool  BufferCreate(
   const int   buffer_index,      // indice buffer
   const uint  size_in_bytes,     // grandezza buffer in bytes
   const uint  flags              // combinazione di flag che definiscono le proprieta
   );
```

Parametri

buffer_index

[in] Indice buffer.

size_in_bytes

[in] Grandezza Buffer in bytes.

flags

[in] Proprietà Buffer impostate utilizzando una combinazione di flags.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

BufferFree

Elimina il in corrispondenza dell'indice specificato.

```
bool  BufferFree(
   const int  buffer_index      // indice buffer
   );
```

Parametri

buffer_index

[in] Indice buffer.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

BufferFromArray

Crea un buffer in corrispondenza dell'indice specificato da un array di valori.

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // indice buffer
   T           &data[],               // array di valori
   const uint  data_array_offset,     // offset nell' array di valori, in bytes
   const uint  data_array_count,      // numero di valori dall'array, da scrivere
   const uint  flags                  // combinazione di flags che definiscono le prop
   );
```

Parametri

buffer_index

[in] Indice buffer.

&data[]

[in] Un array di valori da scrivere nel buffer OpenCL.

data_array_offset

[in] Offset nell'array di valori in byte, da cui la scrittura dei valori inizia.

data_array_count

[in] Il numero di valori da scrivere.

flags

[in] Proprietà Buffer impostate utilizzando una combinazione di flags.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

BufferRead

Legge un buffer di OpenCL all'indice specificato in un array.

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // indice buffer
   T           &data[],               // array di valori
   const uint  cl_buffer_offset,      // offset nel buffer OpenCL, in bytes
   const uint  data_array_offset,     // slittamento negli elementi dell' array
   const uint  data_array_count       // numero di valori dal buffer, da leggere
   );
```

Parametri

buffer_index

[in] Indice buffer.

&data[]

[in] Array per ottenere i valori del buffer OpenCL.

cl_buffer_offset

[in] Offset nel buffer OpenCL in byte, da cui iniziare la lettura dei valori.

data_array_offset

[in] Indice del primo elemento dell'array di valori da scrivere del buffer OpenCL.

data_array_count

[in] Il numero di valori da leggere.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

BufferWrite

Scrive un array di valori nel buffer in corrispondenza dell'indice specificato.

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // indice buffer
   T           &data[],               // array di valori
   const uint  cl_buffer_offset,      // offset nel buffer OpenCL, in bytes
   const uint  data_array_offset,     // slittamento negli elementi dell' array
   const uint  data_array_count       // numero di valori dall'array, da scrivere
   );
```

Parametri

buffer_index

[in] Indice buffer.

&data[]

[in] Un array di valori da scrivere nel buffer OpenCL.

cl_buffer_offset

[in] Offset nel buffer OpenCL in byte, da cui partire per iniziare a scrivere i valori.

data_array_offset

[In] Indice del primo elemento dell'array, a partire dalla quale vengono scritti i valori dell'array nel buffer OpenCL .

data_array_count

[in] Il numero di valori da scrivere.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

Execute

Esegue il programma OpenCL in corrispondenza dell'indice specificato.

```
bool  Execute(
   const int   kernel_index,           // indice del kernel
   const int   work_dim,               // dimensione dello spazio dei tasks
   const uint  &work_offset[],         // offset iniziale nello spazio dei tasks
   const uint  &work_size[]            // il numero totale dei tasks
   );
```

Esegue il kernel OpenCL con l'indice specificato e il numero di tasks nel gruppo locale.

```
bool  Execute(
   const int   kernel_index,           // indice del kernel
   const int   work_dim,               // dimensione dello spazio dei tasks
   const uint  &work_offset[],         // offset iniziale nello spazio dei tasks
   const uint  &work_size[],           // numero totale di tasks
   const uint  &local_work_size[]      // numero di tasks nel gruppo locale
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

work_dim

[in] Grandezza dello spazio tasks.

&work_offset[]

[in] [out] Offset iniziale nello spazio tasks. Passati per riferimento.

&work_size[]

[in][out] La grandezza del sottoinsieme dei tasks . Passati per riferimento.

&local_work_size[]

[in][out] La grandezza del sottoinsieme dei tasks locali nel gruppo. Passati per riferimento.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

GetContext

Restituisce l'handle del contesto OpenCL.

```
int  GetContext();
```

Valore di ritorno

Handle del contesto OpenCL.

GetKernel

Restituisce l'handle dell'oggetto kernel in corrispondenza dell'indice specificato.

```
int  GetKernel(
   const int  kernel_index      // indice del kernel
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

Valore di ritorno

Handle dell'oggetto kernel.

GetKernelName

Restituisce il nome dell'oggetto kernel in corrispondenza dell'indice specificato.

```
string  GetKernelName(
   const int  kernel_index      // indice del kernel
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

Valore di ritorno

Nome dell'oggetto kernel.

GetProgram

Restituisce l'handle del programma OpenCL.

```
int  GetProgram();
```

Valore di ritorno

Handle del programma OpenCL.

Initialize

Inizializza il programma OpenCL.

```
bool  Initialize(
   const string  program,           // handle del programma OpenCL
   const bool    show_log=true      // tiene un log
   );
```

Parametri

program

[in] Handle del programma OpenCL.

show_log=true

[in] Abilita il logging dei messaggi.

Valore di ritorno

Restituisce true, se l'inizializzazione è riuscita. In caso contrario, restituisce false.

KernelCreate

Crea un punto di ingresso nel programma OpenCL in corrispondenza dell'indice specificato.

```
bool  KernelCreate(
   const int     kernel_index,     // indice del kernel
   const string  kernel_name       // nome del kernel
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

kernel_name

[in] Nome dell'oggetto kernel.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

KernelFree

Rimuove una funzione di avvio OpenCL in corrispondenza dell'indice specificato.

```
bool  KernelFree(
   const int  kernel_index      // indice del kernel
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

SetArgument

Imposta un parametro per la funzione OpenCL in corrispondenza dell'indice specificato.

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,     // indice del kernel
   const int  arg_index,        // indice dell'argomento della funzione
   T          value             // codice sorgente
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

arg_index

[in] Indice dell'argomento della funzione.

value

[in] Il valore dell'argomento della funzione.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

SetArgumentBuffer

Imposta un buffer OpenCL come parametro della funzione OpenCL in corrispondenza dell'indice specificato.

```
bool  SetArgumentBuffer(
   const int  kernel_index,     // indice del kernel
   const int  arg_index,        // indice dell'argomento della funzione
   const int  buffer_index      // indice buffer
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

arg_index

[in] Indice dell'argomento della funzione.

buffer_index

[in] Indice buffer.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

SetArgumentLocalMemory

Imposta un parametro nella memoria locale per la funzione OpenCL in corrispondenza dell'indice specificato.

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // indice del kernel
   const int  arg_index,             // indice dell'argomento della funzione
   const int  local_memory_size      // grandezza della memoria locale
   );
```

Parametri

kernel_index

[in] Indice dell'oggetto kernel.

arg_index

[in] Indice dell'argomento della funzione.

local_memory_size

[in] Gandezza della memoria locale.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

SetBuffersCount

Imposta il numero di buffer.

```
bool  SetBuffersCount(
   const int  total_buffers      // numero di buffers
   );
```

Parametri

total_buffers

[in] Il numero totale di buffers.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

SetKernelsCount

Imposta il numero di oggetti kernel.

```
bool  SetKernelsCount(
   const int  total_kernels      // number di kernels
   );
```

Parametri

total_kernels

[in] Il numero totale di kernels.

Valore di ritorno

In caso di esecuzione di successo, restituisce true, altrimenti - false.

Shutdown

Discarica il programma OpenCL.

```
void  Shutdown();
```

Valore di ritorno

Nessun valore di ritorno.

SupportDouble

Controlla se i tipi di dati in virgola mobile sono supportati dal dispositivo.

```
bool  SupportDouble();
```

Valore di ritorno

Restituisce true, se il dispositivo supporta i tipi di dati floating point.

Basic Class CObject

Class CObject è la classe base per la costruzione di una libreria standard MQL5.

Descrizione

La Classe CObject consente a tutti i suoi discendenti di far parte di una lista collegata. Inoltre, sono identificati un certo numero di metodi virtuali per l'ulteriore attuazione nelle classi discendenti.

Dichiarazione

```
class CObject
```

Titolo

```
#include <Object.mqh>
```

Gerarchia di ereditarietà

CObject

Discendenti diretti

CAccountInfo, CArray, CChart, CChartObject, CCurve, CDealInfo, CDictionary_Obj_Double, CDictionary_Obj_Obj, CDictionary_String_Obj, CExpertBase, CFile, CHistoryOrderInfo, CList, COrderInfo, CPositionInfo, CString, CSymbolInfo, CTerminalInfo, CTrade, CTreeNode, CWnd, ICondition, IExpression, IMembershipFunction, INamedValue, IParsableRule

I Metodi della Classe per Gruppi

Attributi	
Prev	Ottiene il valore dell' elemento precedente
Prev	Imposta il valore dell' elemento precedente
Next	Ottiene il valore dell'elemento successivo
Next	Imposta l'elemento successivo
Confronta i metodi	
virtual Compare	Restituisce il risultato del confronto con un altro oggetto
Input/output	
virtual Save	Scrive oggetto in un file
virtual Load	Legge l'oggetto dal file
virtual Type	Restituisce il tipo di oggetto

Prev

Ottiene un puntatore all'elemento precedente nell'elenco.

```
CObject*  Prev()
```

Valore di ritorno

Puntatore all'elemento precedente nell'elenco. Se un elemento è listato per primo, allora restituisce NULL.

Esempio:

```
//--- esempio per CObject::Prev()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Errore creazione oggetto");
      delete object_first;
      return;
     }
   //--- imposta interconnessione
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- usa oggetto prev(precedente)
   CObject *object=object_second.Prev();
   //--- elimina oggetti
   delete object_first;
   delete object_second;
  }
```

Prev

Imposta il puntatore all'elemento precedente nella lista.

```
void  Prev(
   CObject*  object      // Puntatore al precedente elemento nella lista
   )
```

Parametri

object

[in] Nuovo valore del puntatore al prossimo elemento nella lista.

Esempio:

```
//--- esempio per CObject::Prev(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Errore creazione oggetto");
      delete object_first;
      return;
     }
   //--- imposta interconnessione
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- usa oggetti
   //--- ...
   //--- elimina oggetti
   delete object_first;
   delete object_second;
  }
```

Next

Ottiene un puntatore al prossimo elemento della lista.

```
CObject*  Next()
```

Valore di ritorno

Puntatore all'elemento successivo nella lista. Se questo è l'ultimo elemento della lista, restituisce NULL.

Esempio:

```
//--- esempio per CObject::Next()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Errore creazione oggetto");
      delete object_first;
      return;
     }
   //--- imposta interconnessione
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- usa l'oggetto next
   CObject *object=object_first.Next();
   //--- elimina oggetti
   delete object_first;
   delete object_second;
  }
```

Next

Imposta il puntatore al prossimo elemento della lista.

```
void  Next(
   CObject*  object      // Puntatore al prossimo elemento nella lista
   )
```

Parametri

object

[in] Nuovo valore del puntatore all'elemento successivo nella lista.

Esempio:

```
//--- esempio per CObject::Next(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Errore creazione oggetto");
      delete object_first;
      return;
     }
   //--- imposta interconnessione
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- usa oggetti
   //--- ...
   //--- elimina oggetti
   delete object_first;
   delete object_second;
  }
```

Compare

Confronta i dati su un elemento della lista con i dati di un altro elemento di un'altra lista.

```
virtual int  Compare(
   const CObject*  node,      // elemento
   const int       mode=0     // variante
   ) const
```

Parametri

node

[in] Puntatore ad un elemento della lista da confrontare

mode=0

[in] Variante di confronto

Valore di ritorno

0 - nel caso in cui gli elementi della lista sono uguali, -1 - se l'elemento della lista è inferiore all'elemento utilizzato per il confronto (nodo), 1 - se l'elemento della lista è maggiore del elemento utilizzato per il confronto (nodo).

Nota

Il metodo Compare() della classe CObject restituisce sempre 0 e non esegue alcuna azione. Se si desidera confrontare i dati in una classe derivata, dovrebbe essere attuato il metodo Compare(...) . Il parametro 'modalità' deve essere utilizzato in sede di attuazione del confronto multivariato.

Esempio:

```
//--- esempio per CObject::Compare(...)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Errore creazione oggetto");
      delete object_first;
      return;
     }
   //--- imposta interconnessione
```

```
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- confronta oggetti
   int result=object_first.Compare(object_second);
   //--- elimina oggetti
   delete object_first;
   delete object_second;
  }
```

Save

Salva la Lista dati degli elementi da un file.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parametri

file_handle

[in] Handle del file binario aperto in precedenza utilizzando la funzione di FileOpen()

Valore di ritorno

true - completato con successo, false - errore.

Nota

Il metodo Save(int) nella classe CObject restituisce sempre 'true' e non esegue alcuna azione. Se si desidera salvare i dati di una classe derivata in un file, dovrebbe essere attuato il metodo Save(int).

Esempio:

```
//--- esempio per CObject::Save(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- imposta dati oggetto
   //--- . . .
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
```

```
      FileClose(file_handle);
     }
   delete object;
  }
```

Load

Carica la Lista dati degli elementi da un file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario aperto in precedenza utilizzando la funzione FileOpen().

Valore di ritorno

true - completato con successo, false - errore.

Nota

Il metodo Load(int) nella classe CObject restituisce sempre 'true' e non esegue alcuna azione. Se si desidera caricare i dati di una classe derivata da un file, dovrebbe essere attuato il metodo Load(int).

Esempio:

```
//--- esempio per CObject::Load(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   //--- usa oggetto
   //--- . . .
   delete object;
  }
```

Type

Ottiene il tipo d'identificatore.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d'identificatore (per CObject - 0).

Esempio:

```
//--- esempio per CObject::Type()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo d'oggetto
   int type=object.Type();
   //--- elimina oggetto
   delete object;
  }
```

Strutture Dati

Questa sezione contiene i dettagli tecnici su come lavorare con varie strutture di dati (array, liste collegate, ecc) e la descrizione dei componenti rilevanti della libreria standard MQL5.

Utilizzando classi di strutture di dati consente di risparmiare tempo durante la creazione di archivi di dati personalizzati di vari formati (tra cui le strutture di dati compositi).

MQL5 Standard Library (in termini di set di dati) viene inserito nella directory di lavoro del terminale nella cartella \Include\Arrays.

Data Arrays

L'utilizzo di classi di array di dati dinamici consente di risparmiare tempo durante la creazione di archivi di dati personalizzati di vari formati (compresi gli array multidimensionali).

MQL5 standard Library (in termini di array di dati) si trova nella directory di lavoro del terminale nella cartella \Include\Array.

Classe	Descrizione
CArray	Classe base di array di dati dinamici
CArrayChar	Array dinamico di variabili char o uchar
CArrayShort	Array dinamico di variabili short o ushort
CArrayInt	Array dinamico di variabili int o uint
CArrayLong	Array dinamico di variabili long o ulong
CArrayFloat	Array dinamico di variabili float
CArrayDouble	Array dinamico di variabili double
CArrayString	Array dinamico di variabili string
CArrayObj	Array dinamico di puntatori CObject
CList	Offre la possibilità di lavorare con una lista di istanze della classe CObject e le sue discendenti
CTreeNode	Offre la possibilità di lavorare con i nodi dell'albero binario Ctree
CTree	Fornisce la capacità di lavorare con l'albero binario delle istanze della classe CTreeNode e dei suoi discendenti

CArray

La Classe CArray è la classe base di un array dinamico di variabili.

Descrizione

La Classe CArray è destinata ad operare su array dinamico di variabili: allocazione della memoria, ordinamento, e lavorao con i file.

Dichiarazione

```
class CArray : public CObject
```

Titolo

```
#include <Arrays\Array.mqh>
```

Gerarchia di ereditarietà

CObject

CArray

Discendenti diretti

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

I Metodi della Classe per Gruppi

Attributi	
Step	Ottiene la grandezza di incremento dell'array
Step	Imposta la grandezza di incremento dell'array
Totale	Ottiene il numero di elementi nell'array
Disponibile	Ottiene il numero di elementi liberi dell' array che sono disponibili senza allocazione di memoria aggiuntiva
Max	Ottiene la grandezza massima possibile dell'array senza riallocazione di memoria
IsSorted	Ottiene la flag dell'array da ordinare utilizzando la modalità di ordinamento specificato
SortMode	Ottiene la modalità di ordinamento per un array
Metodi Pulisci	
Clear	Elimina tutti gli elementi dell'array, senza rilascio di memoria
Metodi di ordinamento	
Sort	Ordina un array per l'opzione specificata

Attributi	
Input/output	
virtual Save	Salva array di dati in un file
virtual Load	Carica i dati array da un file

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Step

Ottiene la grandezza dell' incremento dell'array .

```
int  Step() const
```

Valore di ritorno

Grandezza Incremento dell'array.

Esempio:

```
//--- esempio per CArray::Step()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene lo step del ridimensionamento
   int step=array.Step();
   //--- usa array
   //--- ...
   //--- elimina array
   delete array;
  }
```

Step

Imposta la dimensione dell'incremento della matrice.

```
bool  Step(
   int  step      // step
   )
```

Parametri

step

[in] Il nuovo valore della grandezza di incremento dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di stabilire uno step minore o uguale a zero.

Esempio:

```
//--- esempio per CArray::Step(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- imposta ridimensionamento step
   bool result=array.Step(1024);
   //--- usa array
   //--- ...
   //--- elimina array
   delete array;
  }
```

Totale

Ottiene il numero di elementi nell'array.

```
int  Total() const;
```

Valore di ritorno

Numero di elementi dell'array.

Esempio:

```
//--- esempio per CArray::Total()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controlla totale
   int total=array.Total();
   //--- usa array
   //--- ...
   //--- elimina array
   delete array;
  }
```

Disponibile

Ottiene il numero di elementi liberi dell'array che sono disponibili senza allocazione di memoria aggiuntiva.

```
int  Available() const
```

Valore di ritorno

Numero di elementi liberi dell'array, disponibili senza allocazione di memoria aggiuntiva.

Esempio:

```
//--- esempio per CArray::Available()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controllo disponibilità
   int available=array.Available();
   //--- usa array
   //--- ...
   //--- elimina array
   delete array;
  }
```

Max

Ottiene la grandezza massima possibile dell' array senza riallocazione di memoria.

```
int  Max() const
```

Valore di ritorno

La dimensione massima possibile dell'array senza riallocazione di memoria.

Esempio:

```
//--- esempio per CArray::Max()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controlla grandezza massima
   int max=array.Max();
   //--- usa array
   //--- ...
   //--- elimina array
   delete array;
  }
```

IsSorted

Ottiene la flag dell'array da ordinare utilizzando la modalità di ordinamento specificato.

```
bool  IsSorted(
   int  mode=0      // modalità di ordinamento
   ) const
```

Parametri

mode=0

[in] Modalità di ordinamento Tested.

Valore di ritorno

Flag della lista ordinata. Se la lista è ordinata utilizzando la modalità specificata - true, altrimenti - false.

Nota

La flag ordinamento non può essere modificata direttamente. Viene impostata dal metodo Sort() e resettata con qualsiasi metodo add/insert tranne per l' InserSort(...).

Esempio:

```
//--- esempio per CArray::IsSorted()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controlla ordinati
   if(array.IsSorted())
     {
      //--- usa metdo per array ordinati
      //--- ...
     }
   //--- elimina array
   delete array;
  }
```

SortMode

Ottiene la modalità di ordinamento per un array.

```
int  SortMode() const;
```

Valore di ritorno

Modalità di ordinamento.

Esempio:

```
//--- esempio per CArray::SortMode()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controlla modalità di ordinamento
   int sort_mode=array.SortMode();
   //--- usa array
   //--- ...
   //--- elimina array
   delete array;
  }
```

Clear

Elimina tutti gli elementi dell'array, senza rilascio di memoria.

```
void  Clear()
```

Valore di ritorno

Nessuno.

Esempio:

```
//--- esempio per CArray::Clear()
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Errore creazione oggetto");
 return;
 }
 //--- usa array
 //--- ...
 //--- pulisci array
 array.Clear();
 //--- elimina array
 delete array;
 }
```

Sort

Ordina un array utilizzando l'opzione specificata.

```
void  Sort(
   int  mode=0      // modalità di ordinamento
   )
```

Parametri

mode=0

[in] Modalità di ordinamento array.

Valore di ritorno

No.

Nota

L'ordinamento di un array è sempre crescente. Per gli array di tipi di dati primitivi (CArrayChar, CArrayShort, ecc), il parametro 'modalità'(mode) non viene utilizzato. Per l'array CArrayObj, l'ordinamento multivariata dovrebbe essere attuato nel metodo Sort(int) delle classi derivate.

Esempio:

```
//--- esempio per CArray::Sort(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ordinamento per modalità 0
   array.Sort(0);
   //--- usa array
   //--- ...
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle di un file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArray::Save(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- elimina array
   delete array;
  }
```

Load

Carica i dati array da un file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle di un file binario precedentemente aperto con la funzione FileOpen().

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArray::Load(...)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- elimina array
   delete array;
  }
```

CArrayChar

CArrayChar è una classe di array dinamico di variabili char o uchar.

Descrizione

La Classe CArrayChar offre la possibilità di lavorare con un array dinamico variabili di char o uchar. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Dichiarazione

```
class CArrayChar : public CArray
```

Titolo

```
#include <Arrays\ArrayChar.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayChar

Metodi della Classe

Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con un rilascio pieno di memoria
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunge elementi di un array alla fine di un altro
AddArray	Aggiunge elementi di un array alla fine di un altro
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro
AssignArray	Copia gli elementi di un array in un altro
Metodi modifica	
Aggiorna	Cambia l'elemento nella posizione specificata dell'array

Controllo della memoria	
Shift	Sposta un elemento da una data posizione nella array per l'offset specificato
Metodi eliminazione	
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
CompareArray	Confronta l'array con un altro
Metodi ordinamento array	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato
SearchLinear	Ricerche l'elemento pari al campione nell'array
Input/output	
virtual Save	Salva array di dati nel file
virtual Load	Carica i dati dell'array dal file
virtual Type	Ottiene l'identificatore tipo array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene incrementata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayChar::Reserve(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riserva memoria
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Ridimensiona

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayChar::Resize(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shutdown

Ripulisce l'array con una rilascio pieno della memoria.

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CArrayChar::Shutdown()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   char  element      // elemento da aggiungere
   )
```

Parametri

element

[in] Valore dell'elemento da aggiungere all' array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Esempio:

```
//--- esempio per CArrayChar::Add(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const char&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array di elementi sorgente da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayChar::AddArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayChar*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayChar utilizzato come sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayChar::AddArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
```

```
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   char  element,     // elemento da inserire
   int   pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayChar::Insert(char,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const char&  src[],     // array sorgente
   int           pos       // posizione
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da inserire

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayChar::InsertArray(const char &[],int)
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   CArrayChar*  src,      // puntatore alla sorgente
   int           pos      // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayChar in utilizzo come sorgente di elementi da inserire.

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayChar::InsertArray(const CArrayChar*,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
```

```
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const char&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayChar::AssignArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const CArrayChar*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayChar in utlizzo come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayChar::AssignArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
   //--- gli arrays sono identici
```

```
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int   pos,          // posizione
   char  element       // valore
   )
```

Parametri

pos

[in] Posizione dell'elemento nell'array da cambiare

element

[in] Nuovo valore dell'elemento

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Esempio:

```
//--- esempio per CArrayChar::Update(int,char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,'A'))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shift

Sposta un elemento da una data posizione nell'array per l'offset specificato.

```
bool  Shift(
   int  pos,        // posizione
   int  shift       // valore
   )
```

Parametri

pos

[in] Posizione dell'elemento mosso nell'array

shift

[in] Il valore di slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayChar::Shift(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Esempio:

```
//--- esempio per CArrayChar::Delete(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina elemento
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione del secondo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non posso rimuovere gli elementi.

Esempio:

```
//--- esempio per CArrayChar::DeleteRange(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
char  At(
   int  pos      // posizione
   ) const
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, CHAR_MAX - c'è stato un tentativo di ottenere un elemento da una posizione non-esistente (l'ultimo codice di errore è ERR_OUT_OF_RANGE).

Nota

Naturalmente, CHAR_MAX può essere un valore valido di un elemento dell'array. Pertanto, controllare sempre l'ultimo codice di errore dopo aver ricevuto un tale valore.

Esempio:

```
//--- esempio per CArrayChar::At(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- errore per la lettura dall'array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
```

```
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const char&  src[]      // array sorgente
   ) const
```

Parametri

src[]

[iin] Riferimento ad un array usato come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayChar::CompareArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const CArrayChar*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayChar in utilizzo come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayChar::CompareArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   char  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayChar::InsertSort(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort('A'))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayChar::Search(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.Search('A')!=-1) printf("Elemento trovato");
   else                      printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato, in caso di successo, -1 - l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayChar::SearchGreat(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreat('A')!=-1) printf("Element found");
   else                           printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayChar::SearchLess(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLess('A')!=-1) printf("Elemento trovato");
   else                          printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

Ricerca un elemento con un valore maggiore o uguale al valore del campione nell' array ordinato.

```
int  SearchGreatOrEqual(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayChar::SearchGreatOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreatOrEqual('A')!=-1) printf("Elemento trovato");
   else                                  printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayChar::SearchLessOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLessOrEqual('A')!=-1) printf("Elemento trovato");
   else                                 printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayChar::SearchFirst(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchFirst('A')!=-1) printf("Elemento trovato");
   else                           printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayChar::SearchLast(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLast('A')!=-1) printf("Elemento trovato");
   else                          printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLinear

Ricerca l'elemento uguale al campione dell'array.

```
int  SearchLinear(
   char  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Nota

Il metodo usa l'algoritmo di ricerca lineare (o ricerca sequenziale) per array non ordinati.

Esempio:

```
//--- esempio per CArrayChar::SearchLinear(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- cerca elemento
   if(array.SearchLinear('A')!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva array di dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayChar::Save(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- elimina array
   delete array;
  }
```

Load

Carica i dati dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayChar::Load(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa elementi dell'array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = '%c'",i,array.At(i));
```

```
      }
   //--- elimina array
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore tipo array (for CArrayChar - 77).

Esempio:

```
//--- esempio per ArrayChar::Type()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CArrayShort

CArrayShort è una classe di array dinamico di variabili short o ushort .

Descrizione

La Classe CArrayShort offre la possibilità di lavorare con un array dinamico di variabili short o ushort. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Dichiarazione

```
class CArrayShort : public CArray
```

Titolo

```
#include <Arrays\ArrayShort.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayShort

I Metodi della Classe per Gruppi

Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con un rilascio pieno di memoria
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunge elementi di un array alla fine di un altro
AddArray	Aggiunge elementi di un array alla fine di un altro
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro
AssignArray	Copia gli elementi di un array in un altro
Update methods	

Controllo della memoria	
Aggiorna	Cambia l'elemento nella posizione specificata dell'array
Shift	Sposta un elemento da una data posizione nella array per l'offset specificato
Metodi eliminazione	
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
CompareArray	Confronta l'array con un altro
Operazioni array ordinato	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato
SearchLinear	Ricerche l'elemento pari al campione nell'array
Input/output	
virtual Save	Salva array di dati nel file
virtual Load	Carica i dati dell'array dal file
virtual Type	Ottiene il tipo identificatore array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata usando lo step precedentemente determinato dal metodo Step(int) o lo step default di 16.

Esempio:

```
//--- esempio per CArrayShort::Reserve(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riserva memoria
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Ridimensiona

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayShort::Resize(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shutdown

Ripulisce l'array con una rilascio pieno della memoria.

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CArrayShort::Shutdown()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   short  element      // elemento da aggiungere
   )
```

Parametri

element

[in] Valore dell'elemento da aggiungere all' array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Esempio:

```
//--- esempio per CArrayShort::Add(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const short&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array di elementi sorgente da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayShort::AddArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayShort*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayShort in utilizzo come sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayShort::AddArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
```

```
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   short  element,     // elemento da inserire
   int    pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayShort::Insert(short,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const short&  src[],     // array sorgente
   int            pos        // posizione
   )
```

Parametri

src[]

[in] Riferimento ad un array in utilizzo come sorgente di elementi da inserire

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayShort::InsertArray(const short &[],int)
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   CArrayShort*  src,      // puntatore alla sorgente
   int            pos      // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayShort in utilizzo come sorgente di elementi da inserire.

pos

[in] Posizione nell'array in cui inserire.

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayShort::InsertArray(const CArrayShort*,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
```

```
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const short&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayShort::AssignArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const CArrayShort*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayShort in utilizzo come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayShort::AssignArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
   //--- gli arrays sono identici
```

```
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int    pos,          // posizione
   short  element       // valore
   )
```

Parametri

pos

[in] Posizione dell'elemento nell'array da cambiare

element

[in] Nuovo valore dell'elemento

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Esempio:

```
//--- esempio per CArrayShort::Update(int,short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,100))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shift

Sposta un elemento da una data posizione nell'array per l'offset specificato.

```
bool  Shift(
   int  pos,        // posizioni
   int  shift       // slittamento
   )
```

Parametri

pos

[in] Posizione dell'elemento mosso nell'array

shift

[in] Il valore di slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayShort::Shift(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Esempio:

```
//--- esempio per CArrayShort::Delete(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina elemento
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione del secondo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non posso rimuovere gli elementi.

Esempio:

```
//--- esempio per CArrayShort::DeleteRange(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
short  At(
   int  pos      // posizione
   ) const
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, SHORT_MAX - c'è stato un tentativo di ottenere un elemento da una posizione non-esistente (l'ultimo codice di errore è ERR_OUT_OF_RANGE).

Nota

Naturalmente, SHORT_MAX può essere un valore valido di un elemento dell'array. Pertanto, controllare sempre l'ultimo codice di errore dopo aver ricevuto un tale valore.

Esempio:

```
//--- esempio per CArrayShort::At(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- errore per la lettura dall'array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
```

```
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const short&  src[]      // array sorgente
   ) const
```

Parametri

src[]

[iin] Riferimento ad un array usato come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayShort::CompareArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const CArrayShort*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayShort in utilizzo come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayShort::CompareArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   short  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayShort::InsertSort(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort(100))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayShort::Search(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.Search(100)!=-1) printf("Elemento trovato");
   else                      printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo, -1 - elemento non trovato.

Esempio:

```
//--- esempio per CArrayShort::SearchGreat(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreat(100)!=-1) printf("Elemento trovato");
   else                           printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayShort::SearchLess(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLess(100)!=-1) printf("Elemento trovato");
   else                          printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

La ricerca di un elemento con un valore maggiore o uguale al valore del campione nel vettore ordinato.

```
int  SearchGreatOrEqual(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayShort::SearchGreatOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreatOrEqual(100)!=-1) printf("Elemento trovato");
   else                                  printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayShort::SearchLessOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLessOrEqual(100)!=-1) printf("Elemento trovato");
   else                                 printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayShort::SearchFirst(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchFirst(100)!=-1) printf("Elemento trovato");
   else                           printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayShort::SearchLast(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLast(100)!=-1) printf("Elemento trovato");
   else                          printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLinear

Cerca un elemento uguale al campione nell'array.

```
int  SearchLinear(
   short  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Nota

Il metodo usa l'algoritmo di ricerca lineare (o ricerca sequenziale) per array non ordinati.

Esempio:

```
//--- esempio per CArrayShort::SearchLinear(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- cerca elemento
   if(array.SearchLinear(100)!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayShort::Save(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Carica i dati array dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayShort::Load(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa elementi dell'array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %d",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Tipo identificatore array (per CArrayShort - 82).

Esempio:

```
//--- esempio per CArrayShort::Type()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CArrayInt

CArrayInt è una classe di array dinamico di variabili int o uint.

Descrizione

La Classe CArrayInt fornisce la possibilità di lavorare con un array dinamico di variabili int o uint. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Dichiarazione

```
class CArrayInt : public CArray
```

Titolo

```
#include <Arrays\ArrayInt.mqh>
```

Gerarchia di ereditarietà

CObject

CArray

CArrayInt

Discendenti diretti

CSpreadBuffer

I Metodi della Classe per Gruppi

Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con un rilascio pieno di memoria
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunge elementi di un array alla fine di un altro
AddArray	Aggiunge elementi di un array alla fine di un altro
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro

Controllo della memoria	
AssignArray	Copia gli elementi di un array in un altro
Update methods	
Aggiorna	Cambia l'elemento nella posizione specificata dell'array
Shift	Sposta un elemento da una data posizione nella array per l'offset specificato
Metodi eliminazione	
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
CompareArray	Confronta l'array con un altro
Operazioni array ordinato	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato
SearchLinear	Ricerche l'elemento pari al campione nell'array
Input/output	
virtual Save	Salva array di dati nel file
virtual Load	Carica i dati dell'array dal file
virtual Type	Ottiene il tipo identificatore array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata usando lo step precedentemente determinato dal metodo Step(int) o lo step default di 16.

Esempio:

```
//--- esempio per CArrayInt::Reserve(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riserva memoria
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Ridimensiona

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // numero
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayInt::Resize(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shutdown

Ripulisce l'array con una rilascio pieno della memoria.

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CArrayInt::Shutdown()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   int  element      // elemento da aggiungere
   )
```

Parametri

element

[in] Valore dell'elemento da aggiungere all' array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Esempio:

```
//--- esempio per CArrayInt::Add(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const int&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array di elementi sorgente da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayInt::AddArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayInt*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayInt in utilizzo come sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayInt::AddArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
```

```
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   int  element,     // elemento da inserire
   int  pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayInt::Insert(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const int&  src[],     // array sorgente
   int          pos        // posizione
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da inserire

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayInt::InsertArray(const int &[],int)
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   CArrayInt*  src,     // puntatore alla sorgente
   int          pos      // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayInt utilizzato come sorgente di elementi da inserire.

pos

[in] Posizione nell'array in cui inserire.

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayInt::InsertArray(const CArrayInt*,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
      delete array;
```

```
      return;
     }
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const int&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayInt::AssignArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const CArrayInt*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayInt utilizzato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayInt::AssignArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
```

```
   //--- gli arrays sono identici
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int  pos,          // posizione
   int  element       // valore
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da cambiare.

element

[in] Nuovo valore dell'elemento

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Esempio:

```
//--- esempio per CArrayInt::Update(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,10000))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shift

Sposta un elemento da una data posizione nell'array per l'offset specificato.

```
bool  Shift(
   int  pos,        // posizione
   int  shift       // slittamento
   )
```

Parametri

pos

[in] Posizione dell'elemento mosso nell'array

shift

[in] Il valore di slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayInt::Shift(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Esempio:

```
//--- esempio per CArrayInt::Delete(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina elemento
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione del secondo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non posso rimuovere gli elementi.

Esempio:

```
//--- esempio per CArrayInt::DeleteRange(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
int  At(
   int  pos      // posizione
   ) const
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, INT_MAX - c'è stato un tentativo di ottenere un elemento da una posizione non-esistente (l'ultimo codice di errore è ERR_OUT_OF_RANGE).

Nota

Naturalmente, INT_MAX può essere un valore valido di un elemento dell'array. Pertanto, controllare sempre l'ultimo codice di errore dopo aver ricevuto un tale valore.

Esempio:

```
//--- esempio per CArrayInt::At(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- errore per la lettura dall'array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
```

```
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const int&  src[]      // array sorgente
   ) const
```

Parametri

src[]

[iin] Riferimento ad un array usato come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayInt::CompareArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const CArrayInt*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayInt utilizzata come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayInt::CompareArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   int  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayInt::InsertSort(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort(10000))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayInt::Search(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.Search(10000)!=-1) printf("Elemento trovato");
   else                        printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayInt::SearchGreat(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreat(10000)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayInt::SearchLess(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLess(10000)!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

La ricerca di un elemento con un valore maggiore o uguale al valore del campione nel vettore ordinato.

```
int  SearchGreatOrEqual(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayInt::SearchGreatOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreatOrEqual(10000)!=-1) printf("Elemento trovato");
   else                                    printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayInt::SearchLessOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLessOrEqual(10000)!=-1) printf("Elemento trovato");
   else                                   printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayInt:: SearchFirst(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchFirst(10000)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayInt::SearchLast(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLast(10000)!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLinear

Ricerca l'elemento uguale al campione dell'array.

```
int  SearchLinear(
   int  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Nota

Il metodo usa l'algoritmo di ricerca lineare (o ricerca sequenziale) per array non ordinati.

Esempio:

```
//--- esempio per CArrayInt::SearchLinear(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- cerca elemento
   if(array.SearchLinear(10000)!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayInt::Save(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Carica i dati array dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayInt::Load(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa elementi dell'array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %d",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore tipo array (per CArrayInt - 82).

Esempio:

```
//--- esempio per CArrayInt::Type()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CArrayLong

La Classe CArrayLong è una classe di array dinamico di variabili long o ulong.

Descrizione

La classe CArrayLong offre la possibilità di lavorare con un array dinamico di variabili long o ulong. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Dichiarazione

```
class CArrayLong : public CArray
```

Titolo

```
#include <Arrays\ArrayLong.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayLong

Discendenti diretti

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

I Metodi della Classe per Gruppi

Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con un rilascio pieno di memoria
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunge elementi di un array alla fine di un altro
AddArray	Aggiunge elementi di un array alla fine di un altro
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro

Controllo della memoria	
AssignArray	Copia gli elementi di un array in un altro
Update methods	
Aggiorna	Cambia l'elemento nella posizione specificata dell'array
Shift	Sposta un elemento da una data posizione nella array per l'offset specificato
Metodi eliminazione	
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
CompareArray	Confronta l'array con un altro
Operazioni array ordinato	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato
SearchLinear	Ricerche l'elemento pari al campione nell'array
Input/output	
virtual Save	Salva array di dati nel file
virtual Load	Carica i dati dell'array dal file
virtual Type	Ottiene il tipo identificatore array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata usando lo step precedentemente determinato dal metodo Step(int) o lo step default di 16.

Esempio:

```
//--- esempio per CArrayLong::Reserve(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riserva memoria
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Resize

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayLong::Resize(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shutdown

Ripulisce l'array con una rilascio pieno della memoria.

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CArrayLong::Shutdown()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   long  element      // elemento da aggiungere
   )
```

Parametri

element

[in] Valore dell'elemento da aggiungere all' array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Esempio:

```
//--- esempio per CArrayLong::Add(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const long&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array di elementi sorgente da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayLong::AddArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayLong*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayLong in utilizzo come sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayLong::AddArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
```

```
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   long  element,     // elemento da inserire
   int   pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayLong::Insert(long,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const long&  src[],     // array sorgente
   int           pos        // posizione
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da inserire

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayLong::InsertArray(const long &[],int)
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   CArrayLong*  src,      // puntatore alla sorgente
   int           pos      // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayLong in utilizzo come sorgente di elementi da inserire.

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayLong::InsertArray(const CArrayLong*,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
```

```
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const long&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayLong::AssignArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const CArrayLong*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayLong utilizzato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayLong::AssignArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
   //--- gli arrays sono identici
```

```
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int   pos,          // posizione
   long  element       // valore
   )
```

Parametri

pos

[in] Posizione dell'elemento nell'array da cambiare

element

[in] Nuovo valore dell'elemento

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Esempio:

```
//--- esempio per CArrayLong::Update(int,long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,1000000))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shift

Sposta un elemento da una data posizione nell'array per l'offset specificato.

```
bool  Shift(
   int  pos,        // posizione
   int  shift       // slittamento
   )
```

Parametri

pos

[in] Posizione dell'elemento mosso nell'array

shift

[in] Il valore di slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayLong::Shift(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Esempio:

```
//--- esempio per CArrayLong::Delete(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina elemento
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione del secondo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non posso rimuovere gli elementi.

Esempio:

```
//--- esempio per CArrayLong::DeleteRange(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
long  At(
   int  pos      // posizione
   ) const
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, LONG_MAX - c'è stato un tentativo di ottenere un elemento da una posizione non-esistente (l'ultimo codice di errore è ERR_OUT_OF_RANGE).

Nota

Naturalmente, LONG_MAX può essere un valore valido di un elemento dell'array. Pertanto, controllare sempre l'ultimo codice di errore dopo aver ricevuto un tale valore.

Esempio:

```
//--- esempio per CArrayLong::At(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Errore di lettura dall'array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
```

```
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const long&  src[]      // array sorgente
   ) const
```

Parametri

src[]

[iin] Riferimento ad un array usato come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayLong::CompareArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina array
   delete array;
  }
```

CompareArrayconst

Confronta l'array con un altro.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayLong in utilizzo come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayLong::CompareArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   long  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayLong::InsertSort(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort(1000000))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayLong::Search(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.Search(1000000)!=-1) printf("Elemento trovato");
   else                          printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayLong::SearchGreat(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreat(1000000)!=-1) printf("Elemento trovato");
   else                               printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayLong::SearchLess(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLess(1000000)!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

La ricerca di un elemento con un valore maggiore o uguale al valore del campione nel vettore ordinato.

```
int  SearchGreatOrEqual(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayLong::SearchGreatOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("Elemento trovato");
   else                                      printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayLong::SearchLessOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLessOrEqual(1000000)!=-1) printf("Elemento trovato");
   else                                     printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayLong::SearchFirst(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchFirst(1000000)!=-1) printf("Elemento trovato");
   else                               printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayLong::SearchLast(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLast(1000000)!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLinear

Ricerca l'elemento uguale al campione dell'array.

```
int  SearchLinear(
   long  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Nota

Il metodo usa l'algoritmo di ricerca lineare (o ricerca sequenziale) per array non ordinati.

Esempio:

```
//--- esempio per CArrayLong::SearchLinear(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- cerca elemento
   if(array.SearchLinear(1000000)!=-1) printf("Elemento trovato");
   else                                printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayLong::Save(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Carica i dati array dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayLong::Load(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa elementi dell'array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %I64",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Tipo identificatore array (per CArrayLong - 84).

Esempio:

```
//--- esempio per CArrayLong::Type()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CArrayFloat

La Classe CArrayFloat è la classe dell'array dinamico di variabili float.

Descrizione

La classe CArrayFloat offre la possibilità di lavoro con un array dinamico di variabili float. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Dichiarazione

```
class CArrayFloat : public CArray
```

Titolo

```
#include <Arrays\ArrayFloat.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayFloat

I Metodi della Classe per Gruppi

Attributi	
Delta	Imposta la tolleranza del confronto
Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con un rilascio pieno di memoria
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunge elementi di un array alla fine di un altro
AddArray	Aggiunge elementi di un array alla fine di un altro
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro

Attributi	
AssignArray	Copia gli elementi di un array in un altro
Update methods	
Aggiorna	Cambia l'elemento nella posizione specificata dell'array
Shift	Sposta un elemento da una data posizione nella array per l'offset specificato
Metodi eliminazione	
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
CompareArray	Confronta l'array con un altro
Operazioni array ordinato	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato
SearchLinear	Ricerche l'elemento pari al campione nell'array
Input/output	
virtual Save	Salva array di dati nel file
virtual Load	Carica i dati dell'array dal file
virtual Type	Ottiene il tipo identificatore array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Delta

Imposta la tolleranza del confronto.

```
void  Delta(
   float  delta      // tolleranza
   )
```

Parametri

delta

[in] Il nuovo valore della tolleranza del confronto.

Valore di ritorno

None

Nota

La tolleranza del confronto viene utilizzata nella ricerca. I valori vengono considerati uguali se la loro differenza è inferiore o uguale alla tolleranza. La tolleranza di default è 0.0.

Esempio:

```
//--- esempio per CArrayFloat::Delta(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- imposta variazione di confronto
   array.Delta(0.001);
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata usando lo step precedentemente determinato dal metodo Step(int) o lo step default di 16.

Esempio:

```
//--- esempio per CArrayFloat::Reserve(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riserva memoria
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Resize

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayFloat::Resize(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shutdown

Ripulisce l'array con una rilascio pieno della memoria.

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CArrayFloat::Shutdown()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   float  element      // elemento da aggiungere
   )
```

Parametri

element

[in] Valore dell'elemento da aggiungere all' array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Esempio:

```
//--- esempio per CArrayFloat::Add(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const float&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array di elementi sorgente da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayFloat::AddArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayFloat*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayFloat in utilizzo come sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayFloat::AddArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
```

```
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   float  element,     // elemento da inserire
   int    pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire in un array

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayFloat::Insert(float,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const float&  src[],     // array sorgente
   int            pos        // posizione
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da inserire

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayFloat::InsertArray(const float &[],int)
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   CArrayFloat*  src,      // puntatore alla sorgente
   int            pos      // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayFloat utilizzato come sorgente di elementi da inserire.

pos

[in] Posizione nell'array da inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayFloat::InsertArray(const CArrayFloat*,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
```

```
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const float&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array usato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayFloat::AssignArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const CArrayFloat*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayFloat utilizzato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayFloat::AssignArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
   //--- gli arrays sono identici
```

```
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int    pos,          // posizione
   float  element       // valore
   )
```

Parametri

pos

[in] Posizione dell'elemento nell'array da cambiare

element

[in] Nuovo valore dell'elemento

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Esempio:

```
//--- esempio per CArrayFloat::Update(int,float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,100.0))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shift

Sposta un elemento da una posizione data nell'array all'offset specificato.

```
bool  Shift(
   int  pos,        // posizione
   int  shift       // slittamento
   )
```

Parametri

pos

[in] Posizione dell'elemento mosso nell'array

shift

[in] Il valore di slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayFloat::Shift(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Esempio:

```
//--- esempio per CArrayFloat::Delete(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina elemento
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione dell'ultimo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non posso rimuovere gli elementi.

Esempio:

```
//--- esempio per CArrayFloat::DeleteRange(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
float  At(
   int  pos      // posizione
   ) const
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, FLT_MAX - c'è stato un tentativo di ottenere un elemento da una posizione non-esistente (l'ultimo codice di errore è ERR_OUT_OF_RANGE).

Nota

Naturalmente, FLT_MAX può essere un valore valido di un elemento dell'array. Pertanto, controllare sempre l'ultimo codice di errore dopo aver ricevuto un tale valore.

Esempio:

```
//--- esempio per CArrayFloat::At(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- errore di lettura dall' array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
```

```
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const float&  src[]      // array sorgente
   ) const
```

Parametri

src[]

[iin] Riferimento ad un array usato come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayFloat::CompareArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina array
   delete array;
  }
```

AssignArrayconst

Confronta l'array con un altro.

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayFloat utilizzato come sorgente di elementi per il confronto.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayFloat::CompareArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   float  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire nell'array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayFloat::InsertSort(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort(100.0))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayFloat::Search(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.Search(100.0)!=-1) printf("Elemento trovato");
   else                        printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayFloat::SearchGreat(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreat(100.0)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayFloat:: SearchLess(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLess(100.0)!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

La ricerca di un elemento con un valore maggiore o uguale al valore del campione nel vettore ordinato.

```
int  SearchGreatOrEqual(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayFloat::SearchGreatOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Elemento trovato");
   else                                    printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayFloat::SearchLessOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Elemento trovato");
   else                                   printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayFloat::SearchFirst(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchFirst(100.0)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayFloat::SearchLast(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLast(100.0)!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLinear

Ricerca l'elemento uguale al campione dell'array.

```
int  SearchLinear(
   float  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento campione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Nota

Il metodo usa l'algoritmo di ricerca lineare (o ricerca sequenziale) per array non ordinati.

Esempio:

```
//--- esempio per CArrayFloat::SearchLinear(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- cerca elemento
   if(array.SearchLinear(100.0)!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayFloat::Save(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Carica i dati array dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayFloat::Load(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa elementi dell'array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %f",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Tipo identificatore array (per CArrayFloat - 87).

Esempio:

```
//--- esempio per CArrayFloat::Type()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CArrayDouble

La Classe CArrayDouble è una classe di array dinamico di variabili double.

Descrizione

La classe CArrayDouble offre la possibilità di lavorare con un array dinamico di variabili double. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Dichiarazione

```
class CArrayDouble : public CArray
```

Titolo

```
#include <Arrays\ArrayDouble.mqh>
```

Gerarchia di ereditarietà

CObject

CArray

CArrayDouble

Discendenti diretti

CDoubleBuffer

I Metodi della Classe per Gruppi

Attributi	
Delta	Impostare la tolleranza di confronto
Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con un rilascio pieno di memoria
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunge elementi di un array alla fine di un altro
AddArray	Aggiunge elementi di un array alla fine di un altro
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata

Attributi	
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro
AssignArray	Copia gli elementi di un array in un altro
Update methods	
Aggiorna	Cambia l'elemento nella posizione specificata dell'array
Shift	Sposta un elemento da una data posizione nella array per l'offset specificato
Metodi eliminazione	
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
CompareArray	Confronta l'array con un altro
Ricerca di min/max	
Minimum	Ottiene l'indice valore più basso nell'intervallo specificato
Maximum	Ottiene l'indice valore più alto nell'intervallo specificato
Operazioni array ordinato	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato

Attributi	
SearchLinear	Ricerche l'elemento pari al campione nell'array
Input/output	
virtual Save	Salva array di dati nel file
virtual Load	Carica i dati dell'array dal file
virtual Type	Ottiene il tipo identificatore array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Delta

Imposta la tolleranza del confronto.

```
void  Delta(
   double  delta      // tolleranza
   )
```

Parametri

delta

[in] Il nuovo valore della tolleranza del confronto.

Valore di ritorno

No

Nota

La tolleranza del confronto viene utilizzata nella ricerca. I valori vengono considerati uguali se la loro differenza è inferiore o uguale alla tolleranza. La tolleranza di default è 0.0.

Esempio:

```
//--- esempio per CArrayDouble::Delta(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- imposta variazione di confronto
   array.Delta(0.001);
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata usando lo step precedentemente determinato dal metodo Step(int) o lo step default di 16.

Esempio:

```
//--- esempio per CArrayDouble::Reserve(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riserva memoria
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Resize

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayDouble::Resize(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shutdown

Ripulisce l'array con una rilascio pieno della memoria.

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CArrayDouble::Shutdown()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   double  element      // elemento da aggiungere
   )
```

Parametri

element

[in] Valore dell'elemento da aggiungere nell'array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Esempio:

```
//--- esempio per CArrayDouble::Add(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const double&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array di elementi sorgenti da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayDouble::AddArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayDouble*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayDouble utilizzata come sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayDouble::AddArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
```

```
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   double  element,     // elemento da inserire
   int     pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire in un array

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayDouble::Insert(double,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const double&  src[],     // array sorgente
   int            pos        // posizione
   )
```

Parametri

src[]

[in] Riferimento ad un array in utilizzo come sorgente di elementi da inserire

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayDouble::InsertArray(const double &[],int)
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   CArrayDouble*  src,      // puntatore alla sorgente
   int            pos       // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayDouble utilizzato come sorgente di elementi da inserire.

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayDouble::InsertArray(const CArrayDouble*,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
```

```
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const double&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad unarray usato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayDouble::AssignArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const CArrayDouble*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayDouble in utlizzo come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayDouble::AssignArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
   //--- gli arrays sono identici
```

```
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int     pos,          // posizione
   double  element       // valore
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da cambiare

element

[in] Nuovo valore dell'elemento.

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Esempio:

```
//--- esempio per CArrayDouble::Update(int,double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,100.0))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shift

Sposta un elemento da una posizione data nell'array all'offset specificato.

```
bool  Shift(
   int  pos,        // posizione
   int  shift       // slittamento
   )
```

Parametri

pos

[in] Posizione dell'elemento spostato nell'array

shift

[in] Il valore dello slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayDouble::Shift(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Esempio:

```
//--- esempio per CArrayDouble::Delete(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina elemento
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione del secondo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non posso rimuovere gli elementi.

Esempio:

```
//--- esempio per CArrayDouble::DeleteRange(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
double  At(
   int  pos      // posizione
   ) const
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, DBL_MAX - c'è stato un tentativo di ottenere un elemento da una posizione non-esistente (l'ultimo codice di errore è ERR_OUT_OF_RANGE).

Nota

Naturalmente, DBL_MAX può essere un valore valido di un elemento dell'array. Pertanto, controllare sempre l'ultimo codice di errore dopo aver ricevuto un tale valore.

Esempio:

```
//--- esempio per CArrayDouble::At(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Errore di lettura dall'array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
```

```
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const double&  src[]      // array sorgente
   ) const
```

Parametri

src[]

[in] Riferimento ad unarray usato come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayDouble::CompareArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const CArrayDouble*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayDouble utilizzata come sorgente di elementi per il confronto.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayDouble::CompareArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

Minimum

Ottiene l' indice dell'elemento più basso dell'array nell'intervallo specificato.

```
int  Minimum(
   int  start,      // indice di partenza
   int  count       // numero di elementi
   ) const
```

Parametri

start

[in] Indice d'inizio nel range di ricerca.

count

[in] Grandezza del reange di ricerca (numero di elementi).

Valore di ritorno

Indice dell'elemento più basso nell'intervallo specificato.

Maximum

Ottiene l'indice del più alto elemento dell' array nell'intervallo specificato.

```
int  Maximum(
   int  start,      // indice di partenza
   int  count       // numero di elementi
   ) const
```

Parametri

start

[in] Indice d'inizio nel range di ricerca.

count

[in] Grandezza del reange di ricerca (numero di elementi).

Valore di ritorno

Indice dell'elemento più alto nell'intervallo specificato.

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   double  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire in un array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayDouble::InsertSort(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort(100.0))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayDouble::Search(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.Search(100.0)!=-1) printf("Elemento trovato");
   else                        printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayDouble::SearchGreat(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreat(100.0)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayDouble:: SearchLess(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLess(100.0)!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

La ricerca di un elemento con un valore maggiore o uguale al valore del campione nel vettore ordinato.

```
int  SearchGreatOrEqual(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayDouble::SearchGreatOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Elemento trovato");
   else                                    printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayDouble::SearchLessOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Elemento trovato");
   else                                   printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayDouble::SearchFirst(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchFirst(100.0)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayDouble::SearchLast(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLast(100.0)!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLinear

Ricerca l'elemento uguale al campione dell'array.

```
int  SearchLinear(
   double  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Nota

Il metodo usa l'algoritmo di ricerca lineare (o ricerca sequenziale) per array non ordinati.

Esempio:

```
//--- esempio per CArrayDouble::SearchLinear(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- cerca elemento
   if(array.SearchLinear(100.0)!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayDouble::Save(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi array
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   //--- elimina array
   delete array;
  }
```

Load

Carica i dati array dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayDouble::Load(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa elementi dell'array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = %f",i,array.At(i));
```

```
      }
   //--- elimina array
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di identificatore Array (per CArrayDouble - 87).

Esempio:

```
//--- esempio per CArrayDouble::Type()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CArrayString

La classe CArrayString è una classe di array dinamico di variabili string.

Descrizione

La classe CArrayString offre la possibilità di lavorare con un array dinamico di variabili stringa. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Dichiarazione

```
class CArrayString : public CArray
```

Titolo

```
#include <Arrays\ArrayString.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayString

I Metodi della Classe per Gruppi

Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con un rilascio pieno di memoria
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunge elementi di un array alla fine di un altro
AddArray	Aggiunge elementi di un array alla fine di un altro
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro
AssignArray	Copia gli elementi di un array in un altro
Update methods	

Controllo della memoria	
Aggiorna	Cambia l'elemento alla posizione array specificata
Shift	Sposta un elemento da una data posizione nell'array per l'offset specificato
Metodi eliminazione	
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
CompareArray	Confronta l'array con un altro
Operazioni array ordinato	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato
SearchLinear	Ricerche l'elemento pari al campione nell'array
Input/output	
virtual Save	Salva array di dati nel file
virtual Load	Carica i dati dell'array dal file
virtual Type	Ottiene il tipo identificatore array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata usando lo step precedentemente determinato dal metodo Step(int) o lo step default di 16.

Esempio:

```
//--- esempio per CArrayString::Reserve(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riserva memoria
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Resize

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata dello step precedentemente determinato secondo il metodo Step(int) o lo step predefinito di 16.

Esempio:

```
//--- esempio per CArrayString::Resize(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shutdown

Ripulisce l'array con una rilascio pieno della memoria.

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CArrayString::Shutdown()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   string  element      // elemento da aggiungere
   )
```

Parametri

element

[in] Valore dell'elemento da aggiungere nell'array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Esempio:

```
//--- esempio per CArrayString::Add(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const string&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad un array di elementi sorgenti da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayString::AddArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayString*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayString in utilizzo come sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Esempio:

```
//--- esempio per CArrayString::AddArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- aggiunge un altro array
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
```

```
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   string  element,     // elemento da inserire
   int     pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire in un array

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayString::Insert(string,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const string&  src[],     // array sorgente
   int             pos        // posizione
   )
```

Parametri

src[]

[in] Riferimento ad un array in utilizzo come sorgente di elementi da inserire

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayString::InsertArray(const string &[],int)
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   CArrayString*  src,      // puntatore alla sorgente
   int             pos      // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayString in utilizzo come sorgente di elementi da inserire.

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Esempio:

```
//--- esempio per CArrayString::InsertArray(const CArrayString*,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
```

```
      delete array;
      return;
     }
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const string&  src[]      // array sorgente
   )
```

Parametri

src[]

[in] Riferimento ad unarray usato come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayString::AssignArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Copia gli elementi di un array in un altro.

```
bool  AssignArray(
   const CArrayString*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayString in utilizzo come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayString::AssignArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- assegna un altro array
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
   //--- gli arrays sono identici
```

```
   //--- elimina la sorgente dell'array
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int     pos,          // posizione
   string  element       // valore
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da cambiare

element

[in] Nuovo valore dell'elemento

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Esempio:

```
//--- esempio per CArrayString::Update(int, string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,"ABC"))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Shift

Sposta un elemento da una posizione data nell'array all'offset specificato.

```
bool  Shift(
   int  pos,        // posizione
   int  shift       // slittamento
   )
```

Parametri

pos

[in] Posizione dell'elemento spostato nell'array

shift

[in] Il valore dello slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayString::Shift(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Esempio:

```
//--- esempio per CArrayString::Delete(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina elemento
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione dell'ultimo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non posso rimuovere gli elementi.

Esempio:

```
//--- esempio per CArrayString::DeleteRange(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
string  At(
   int  pos      // posizione
   ) const
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, "" c'è stato un tentativo di ottenere un elemento da una posizione non-esistente (l'ultimo codice di errore è ERR_OUT_OF_RANGE).

Nota

Naturalmente, "" può essere un valore valido di un elemento di un array. Pertanto, controllare sempre l'ultimo codice di errore dopo aver ricevuto un tale valore.

Esempio:

```
//--- esempio per CArrayString::At(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result=="" && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Errore di lettura dall'array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
```

```
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const string&  src[]      // array sorgente
   ) const
```

Parametri

src[]

[in] Riferimento ad unarray usato come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali, false - gli array sono diversi.

Esempio:

```
//--- esempio per CArrayString::CompareArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina array
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArrays(
   const CArrayString*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayString in utilizzo come sorgente di elementi per il confronto.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Esempio:

```
//--- esempio per CArrayString::CompareArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- aggiunge gli elementi dell'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   string  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire in un array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Esempio:

```
//--- esempio per CArrayString::InsertSort(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort("ABC"))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayString::Search(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.Search("ABC")!=-1) printf("Elemento trovato");
   else                        printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayString::SearchGreat(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreat("ABC")!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayString:: SearchLess(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLess("ABC")!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

La ricerca di un elemento con un valore maggiore o uguale al valore del campione nel vettore ordinato.

```
int  SearchGreatOrEqual(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayString:: SearchGreatOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("Elemento trovato");
   else                                    printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayString:: SearchLessOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLessOrEqual("ABC")!=-1) printf("Elemento trovato");
   else                                   printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayString:: SearchFirst(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchFirst("ABC")!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayString:: SearchLast(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- cerca elemento
   if(array.SearchLast("ABC")!=-1) printf("Elemento trovato");
   else                            printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLinear

Ricerca l'elemento uguale al campione dell'array.

```
int  SearchLinear(
   string  element      // campione
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Nota

Il metodo usa l'algoritmo di ricerca lineare (o ricerca sequenziale) per array non ordinati.

Esempio:

```
//--- esempio per CArrayString::SearchLinear(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- cerca elemento
   if(array.SearchLinear("ABC")!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayString::Save(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi array
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Carica i dati array dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayString::Load(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa elementi dell'array
   for(int i=0;i<array.Total();i++)
     {
      printf("Elemento[%d] = '%s'",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Tipo identificatore array (per CArrayString - 89).

Esempio:

```
//--- esempio per CArrayString::Type()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CArrayObj

La Classe CArrayObj è una classe di array dinamico di puntatori ad istanze di CObject e delle sue classi derivate.

Descrizione

La Classe CArrayObj offre la possibilità di lavorare con un array dinamico di puntatori a istanze di CObject e di sue classi derivate. Questo permette di lavorare sia con array dinamici multidimensionali di tipi di dati primitivi e con strutture di dati che hanno un'organizzazione più complessa di dati.

La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Ci sono certe sottigliezze della classe CArrayObj.

Dichiarazione

```
class CArrayObj : public CArray
```

Titolo

```
#include <Arrays\ArrayObj.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayObj

Discendenti diretti

CIndicators, CSeries

I Metodi della Classe per Gruppi

Attributi	
FreeMode	Ottiene la flag della gestione della memoria
FreeMode	Imposta la flag di gestione della memoria
Controllo della memoria	
Reserve	Alloca memoria per aumentare la grandezza dell'array
Ridimensiona	Imposta una nuova (più piccolo) grandezza dell'array
Shutdown	Cancella l'array con piena deallocazione della sua memoria (ma non i suoi elementi).
Creazione di un nuovo elemento	

Attributi	
virtual CreateElement	Crea un nuovo elemento di un array nella posizione specificata
Aggiunta metodi	
Add	Aggiunta elemento alla fine dell'array
AddArray	Aggiunta elemento alla fine dell'array
Insert	Inserisce un elemento alla posizione specificata nell array
InsertArray	Inserisce in un array elementi di un altro array dalla posizione specificata
AssignArray	Copia gli elementi di un array in un altro
Update methods	
Aggiorna	Cambia l'elemento alla posizione array specificata
Shift	Sposta un elemento da una data posizione nell'array per l'offset specificato
Metodi eliminazione	
Detach	Ottiene l'elemento dalla posizione specificata e lo rimuove dall'array
Delete	Rimuove l'elemento dalla posizione specificata dell'arrat
DeleteRange	Elimina un gruppo di elementi dalla posizione specificata dell'array
Clear	Rimuovere tutti gli elementi dell'array senza il rilascio della memoria array
Metodi d'accesso	
At	Ottiene l'elemento dalla posizione specificata dell'array
Confronta i metodi	
CompareArray	Confronta l'array con un altro
Operazioni array ordinato	
InsertSort	Inserisce un elemento in un array ordinato
Ricerca	Cerca un elemento uguale al campione(al sample) in un array ordinato
SearchGreat	Cerca un elemento con un valore superiore al valore del campione(il sample) in un array ordinato

Attributi	
SearchLess	Ricerca di un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato
SearchGreatOrEqual	Ricerca di un elemento con un valore maggiore o uguale al valore del campione nell'array ordinato
SearchLessOrEqual	Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato
SearchFirst	Ricerche il primo elemento pari al campione nell' array ordinato
SearchLast	Ricerche l'ultimo elemento pari al campione nell'array ordinato
Input/output	
Save	Salva array di dati nel file
Load	Carica i dati dell'array dal file
Type	Ottiene il tipo identificatore array

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Gli array della classe CObject hanno applicazione pratica (comprese tutte le classi della libreria standard).

Ad esempio, considerare le opzioni per array bidimensionali:

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- crea array
   CArrayObj    *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
```

```
      return;
     }
//--- crea subarrays
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("Errore creazione oggetto");
         return;
        }
      //--- riempie array
      for(j=0;j<second_size;j++)
        {
         sub_array.Add(i*j);
        }
      array.Add(sub_array);
     }
//--- crea array OK
   for(i=0;i<first_size;i++)
     {
      sub_array=array.At(i);
      for(j=0;j<second_size;j++)
        {
         double element=sub_array.At(j);
         //--- use elementi array
        }
     }
   delete array;
  }
```

Subtleties

La classe ha un meccanismo di controllo della memoria dinamica, quindi state attenti quando si lavora con elementi dell'array.

Il meccanismo di gestione della memoria può essere attivato/disattivato con il metodo FreeMode (bool). Per impostazione predefinita, il meccanismo è attivato.

Di conseguenza, ci sono due opzioni per trattare con la classe CArrayObj:

1. Meccanismo di gestione della memoria è abilitata. (default)

In questo caso, CArrayObj assume responsabilità di rilasciare la memoria utilizzata per gli elementi dopo la loro rimozione dall'array. Un programma personalizzato non dovrebbe rilasciare gli elementi dell'array.

Esempio:

```
   int i;
//--- crea un array
   CArrayObj *array=new CArrayObj;
//--- riempie elementi dell' array
   for(i=0;i<10;i++) array.Add(new CObject);
//--- fa qualcosa
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- azioni eseguite con l'elemento
      . . .
     }
//--- rimuove l'array con gli elementi
   delete array;
```

2. Il meccanismo di gestione della memoria è disabilitato.

In questo caso, CArrayObj non è responsabile per deallocazione della memoria degli elementi dopo la loro rimozione dall' array. Inoltre, il programma utente deve rilasciare gli elementi dell'array.

Esempio:

```
   int i;
//--- crea un array
   CArrayObj *array=new CArrayObj;
//--- disabilita il meccanismo di gestione della memoria
   array.FreeMode(false);
//--- riempie array con elementi
   for(i=0;i<10;i++) array.Add(new CObject);
//--- fa qualcosa
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- azioni eseguite con l'elemento
      . . .
     }
//--- rimuove elementi array
   while(array.Total()) delete array.Detach();
//--- rimuove array vuoto
   delete array;
```

FreeMode

Ottiene la flag di della gestione della memoria.

```
bool  FreeMode() const
```

Valore di ritorno

Flag della gestione della memoria.

Esempio:

```
//--- esempio per CArrayObj::FreeMode()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene la flag mode free
   bool array_free_mode=array.FreeMode();
   //--- elimina array
   delete array;
  }
```

FreeMode

Imposta il flag di gestione della memoria.

```
void  FreeMode(
   bool  mode      // nuovo flag
   )
```

Parametri

mode

[in] Nuovo valore del flag di gestione della memoria.

Valore di ritorno

Nessuno.

Nota

L' impostare la flag di gestione della memoria è una parte importante della classe di utilizzo CArrayObj. Dal momento che gli elementi dell'array sono puntatori a oggetti dinamici, è importante determinare che cosa fare con loro durante la rimozione dall'array.

Se il flag è impostato, la rimozione di un elemento dall'array, l'elemento viene cancellato automaticamente dall'operatore delete. Se il flag non è impostato, si presume che un puntatore all'oggetto eliminato è ancora da qualche parte nel programma utente e verrà deallocato dal programma più in la.

Se il programma utente ripristina il flag di gestione della memoria, l'utente deve capire la sua responsabilità per la rimozione dell'array prima della fine del programma. Altrimenti la memoria allocata per gli elementi del nuovo operatore non viene rilasciata.

Per grandi quantità di dati questo potrebbe portare anche ad un crash del vostro terminale.

Se l'utente non azzera il flag di gestione della memoria, c'è un altro trabocchetto. Quando gli elementi puntatore a matrice vengono memorizzati da qualche parte nelle variabili locali, allora la rimuozione dell'array porterà ad un errore critico e crash del programma utente. Per default, viene impostato il flag di gestione della memoria, cioè la classe dell'array è responsabile di liberare gli elementi di memoria.

Esempio:

```
//--- esempio per CArrayObj::FreeMode(bool)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
```

```
   //--- resetta la flag free mode
   array.FreeMode(false);
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Reserve

Alloca la memoria per incrementare la grandezza dell'array.

```
bool  Reserve(
   int  size      // numero
   )
```

Parametri

size

[in] Il numero di elementi aggiuntivi dell'array.

Valore di ritorno

true in caso di successo, false - se vi era un tentativo di richiedere un ammontare minore o uguale a zero, o fallimento nell'incrementare l'array

Nota

Per ridurre la frammentazione della memoria, la grandezza dell'array viene cambiata usando lo step precedentemente determinato dal metodo Step(int) o lo step default di 16.

Esempio:

```
//--- esempio per CArrayObj::Reserve(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("Riserva errore");
      delete array;
      return;
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Resize

Imposta una nuova (più piccola) grandezza dell'array.

```
bool  Resize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Nuova grandezza dell'array.

Valore di ritorno

true - successo, false - c'è stato un tentativo di impostare la grandezza inferiore a zero.

Nota

La modifica della grandezza dell' array consente di utilizzare la memoria in modo ottimale. Gli elementi eccessivi sulla destra vengono perduti. La memoria degli elementi persi viene rilasciata o meno a seconda della modalità di gestione della memoria.

Per ridurre la frammentazione della memoria, la modifica della grandezza dell'array è realizzata con uno step precedentemente fornito attraverso il metodo di Step(int), o 16 (default).

Esempio:

```
//--- esempio per CArrayObj::Resize(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ridimensiona array
   if(!array.Resize(10))
     {
      printf("Ridimensiona errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Clear

Rimuovere tutti gli elementi dell'array senza il rilascio della memoria array.

```
void  Clear()
```

Valore di ritorno

No.

Nota

Se è attivato il flag di gestione della memoria, la memoria utilizzata per gli oggetti eliminati viene rilasciata.

Esempio:

```
//--- esempio per CArrayObj::Clear()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- pulisce array
   array.Clear();
   //--- elimina array
   delete array;
  }
```

Shutdown

Cancella l'array con piena deallocazione di memoria per esso (ma non per i suoi elementi).

```
bool  Shutdown()
```

Valore di ritorno

true - successo, false - è avvenuto un errore.

Nota

Se la gestione della memoria è attivata, la memoria degli elementi eliminati viene deallocata.

Esempio:

```
//--- esempio per CArrayObj::Shutdown()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Errorde dello Shutdown");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

CreateElement

Crea un nuovo elemento dell'array nella posizione specificata.

```
bool  CreateElement(
   int  index      // posizione
   )
```

Parametri

index

[in] Posizione in cui si desidera creare un nuovo elemento.

Valore di ritorno

true - successo, false - non posso creare un elemento.

Nota

Metodo CreateElement(int) nella classe CArrayObj restituisce sempre false e non esegue alcuna azione. Se necessario, il metodo CreateElement(int) dovrebbe essere attuato in una classe derivata.

Esempio:

```
//--- esempio per CArrayObj::CreateElement(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riempie array
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("Errore creazione elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
```

```
}
```

Add

Aggiunge un elemento alla fine dell'array.

```
bool  Add(
   CObject*  element      // elemento da aggiungere
   )
```

Parametri

element

[in] valore dell'elemento da aggiungere all' array.

Valore di ritorno

true - successo, false - non posso aggiungere l'elemento.

Nota

L' elemento non viene aggiunto all' arrat se un puntatore non valido (ad esempio NULL) viene passato come parametro.

Esempio:

```
//--- esempio per CArrayObj::Add(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi array
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("Errore aggiunta elemento");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AddArray

Aggiunge elementi di un array alla fine di un altro.

```
bool  AddArray(
   const CArrayObj *  src      // puntatore alla sorgente array
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayDouble - sorgente di elementi da aggiungere.

Valore di ritorno

true - successo, false - non si possono aggiungere elementi.

Nota

Aggiunta di elementi da array ad array è in realtà la copia dei puntatori. Pertanto, quando si chiama il metodo, c'è un trabocchetto - ci può essere un puntatore a un oggetto dinamico in più di una variabile.

```
//--- esempio
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- crea nuova istanza CArrayObj
//--- il mamagement della memoria default è ON
CArrayObj *array=new CArrayObj;
//--- aggiunta (copia) degli elementi dall'array sorgente
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- esegue azioni erronee
   switch(error)
     {
      case 0:
         //--- rimuove l'array sorgente senza controllare la flag del management della
         delete src;
         //--- risultato:
         //--- è possibile indirizzare un elemento per puntatore non valito nell'array
         break;
      case 1:
         //--- disabilita il meccanismo del management della memoria nell'array sorgen
         if(src.FreeMode()) src.FreeMode(false);
         //--- ma non rimuove l'array sorgente
         //--- risultato:
         //--- dopo aver rimosso l'array ricevente, è possibile indirizzare un element
         break;
```

```
      case 2:
         //--- disabilita il meccanismo del management della memoria nell'array sorgen
         src.FreeMode(false);
         //--- disabilita il meccanismo di management della memoria nell'array ricever
         array.FreeMode(false);
         //--- risultato:
         //--- dopo la terminazione del programma, ottiene un "memory leak" (perdita d
         break;
      }
  }
else
  {
   //--- disabilita il meccanismo di management della memoria nell'array sorgente
   if(src.FreeMode()) src.FreeMode(false);
   //--- elimina l'array sorgente
   delete src;
   //--- result:
   //--- l'indirizzamento dell'array ricevente sarà corretto
   //--- l'eliminazione dell'array ricevente porterà all'eliminazione dei suoi element
  }
```

Esempio:

```
//--- esempio per CArrayObj::AddArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- resetta la flag free mode
   src.FreeMode(false);
   //--- riempie l'array sorgente
   //--- . . .
   //--- aggiunge un altro array
```

```
   if(!array.AddArray(src))
     {
      printf("Errorre aggiuta array");
      delete src;
      delete array;
      return;
     }
   //--- elimina l'array sorgente senza elementi
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Insert

Inserisce un elemento nella posizione specificata dell'array.

```
bool  Insert(
   CObject*  element,     // elemento da inserire
   int       pos          // posizione
   )
```

Parametri

element

[in] Valore dell'elemento da inserire in un array

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Nota

L' elemento non viene aggiunto all' arrat se un puntatore non valido (ad esempio NULL) viene passato come parametro.

Esempio:

```
//--- esempio per CArrayObj::Insert(CObject*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce elementi
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("Inserisci errore");
         delete array;
         return;
        }
     }
   //--- usa array
   //--- . . .
```

```
   //--- elimina array
   delete array;
  }
```

InsertArray

Inserisce elementi di un array dalla posizione specificata di un altro array.

```
bool  InsertArray(
   const CArrayObj*  src,     // puntatore alla sorgente
   int               pos      // posizione
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayObj in utilizzo come sorgente di elementi da inserire.

pos

[in] Posizione nell' array in cui inserire

Valore di ritorno

true - successo, false - non posso inserire gli elementi.

Nota

Vedi: CArrayObj::AddArray(const CArrayObj*).

Esempio:

```
//--- esempio per CArrayObj::InsertArray(const CArrayObj*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- resetta la flag free mode
   src.FreeMode(false);
   //--- riempie l'array sorgente
   //--- . . .
```

```
   //--- inserisce un altro array
   if(!array.InsertArray(src,0))
     {
      printf("Errore inserimento array");
      delete src;
      delete array;
      return;
     }
   //--- elimina l'array sorgente senza elementi
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

AssignArray

Confronta l'array con un altro.

```
bool  AssignArray(
   const CArrayObj*  src      // puntatore alla sorgente
   )
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayObj in utilizzo come sorgente di elementi da copiare.

Valore di ritorno

true - successo, false - non posso copiare gli elementi.

Nota

Se l'array ricevente non è vuoto quando si chiama AssignArray, allora tutti gli elementi saranno rimossi; e se è impostato il flag di gestione della memoria, la memoria utilizzata per gli elementi eliminati verrà rilasciata. L'array ricevente diventa una copia esatta di quello sorgente. In più, vedere CArrayObj::AddArray(const CArrayObj*).

Esempio:

```
//--- esempio per CArrayObj::AssignArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- resetta la flag free mode
   src.FreeMode(false);
   //--- riempie l'array sorgente
   //--- . . .
   //--- assegna un altro array
```

```
   if(!array.AssignArray(src))
     {
      printf("Errore assegnazione array");
      delete src;
      delete array;
      return;
     }
   //--- gli arrays sono identici
   //--- elimina l'array sorgente senza elementi
   delete src;
   //--- usa array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Aggiorna

Cambia l'elemento nella posizione array specificata.

```
bool  Update(
   int       pos,          // posizione
   CObject*  element       // valore
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da cambiare

element

[in] Nuovo valore dell'elemento

Valore di ritorno

true - successo, false - non posso cambiare l'elemento.

Nota

L'elemento non cambia se un puntatore non valido (per esempio, NULL) viene passato come parametro. Se la gestione della memoria è attivata, la memoria di un elemento sostituito è deallocata.

Esempio:

```
//--- esempio per CArrayObj::Update(int,CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- aggiorna elemento
   if(!array.Update(0,new CObject))
     {
      printf("Errore aggiornamento");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
```

```
}
```

Shift

Sposta un elemento da una posizione data nell'array all'offset specificato.

```
bool  Shift(
   int  pos,        // posizione
   int  shift       // slittamento
   )
```

Parametri

pos

[in] Posizione dell'elemento spostato nell'array

shift

[in] Il valore dello slittamento (sia positivo che negativo).

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CArrayObj::Shift(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- slitta elementi
   if(!array.Shift(10,-5))
     {
      printf("Slitta errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Detach

Rimuove un elemento da una data posizione nell'array.

```
CObject*  Detach(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione di un elemento rimosso nell'array.

Valore di ritorno

Puntatore all'elemento rimosso - successo, NULL - non è possibile rimuovere l'elemento.

Nota

Quando un elemento viene rimosso dall'array, non viene eliminato indipendentemente dalla flag di gestione della memoria. Una volta che il puntatore elemento dell'array viene usato, deve essere deallocato.

Esempio:

```
//--- esempio per CArrayObj::Detach(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("Errore distaccamento");
      delete array;
      return;
     }
   //--- usa elemento array
   //--- . . .
   //--- elimina elemento
   delete object;
   //--- elimina array
   delete array;
```

```
}
```

Delete

Rimuove l'elemento dalla posizione specificata dell'array .

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione del elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Nota

Se la gestione della memoria è attivata, la memoria degli elementi eliminati viene deallocata.

Esempio:

```
//--- esempio per CArrayObj::Delete(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   if(!array.Delete(0))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

DeleteRange

Elimina un gruppo di elementi dalla posizione specificata dell'array.

```
bool  DeleteRange(
   int  from,     // posizione del primo elemento
   int  to        // posizione dell'ultimo elemento
   )
```

Parametri

from

[in] Posizione del primo elemento dell'array da rimuovere.

to

[in] posizione dell'ultimo elemento dell'array da rimuovere.

Valore di ritorno

true - successo, false - non ri possono rimuovere gli elementi.

Nota

Se la gestione della memoria è attivata, la memoria degli elementi eliminati viene deallocata.

Esempio:

```
//--- esempio per CArrayObj::DeleteRange(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- elimina gli elementi
   if(!array.DeleteRange(0,10))
     {
      printf("Elimina errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

At

Ottiene l'elemento dalla posizione specificata dell'array.

```
CObject*  At(
   int  pos      // posizione
   )
```

Parametri

pos

[in] Posizione dell'elemento desiderato nell'array.

Valore di ritorno

Il valore dell'elemento - successo, NULL- si è tentato di ottenere un elemento di una posizione inesistente.

Esempio:

```
//--- esempio per CArrayObj::At(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- Errore di lettura dall'array
         printf("Ottiene l'elemento dell'array");
         delete array;
         return;
        }
      //--- usa elemento
      //--- . . .
     }
   delete array;
  }
```

CompareArray

Confronta l'array con un altro.

```
bool  CompareArray(
   const CArrayObj*  src      // puntatore alla sorgente
   ) const
```

Parametri

src

[in] Puntatore ad un'istanza della classe CArrayObj in utilizzo come sorgente di elementi per il confronto.

Valore di ritorno

true - gli array sono uguali - gli array non sono uguali.

Esempio:

```
//--- esempio per CArrayObj::CompareArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea l'array sorgente
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete array;
      return;
     }
   //--- riempie l'array sorgente
   //--- . . .
   //--- confronta con un altro array
   int result=array.CompareArray(src);
   //--- elimina gli arrays
   delete src;
   delete array;
  }
```

InsertSort

Inserisce un elemento in un array ordinato.

```
bool  InsertSort(
   CObject*  element      // elemento da inserire
   )
```

Parametri

element

[in] Valore dell'elemento da inserire in un array ordinato

Valore di ritorno

true - successo, false - non posso inserire l'elemento.

Nota

L' elemento non viene aggiunto all' arrat se un puntatore non valido (ad esempio NULL) viene passato come parametro.

Esempio:

```
//--- esempio per CArrayObj::InsertSort(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- inserisci elemento
   if(!array.InsertSort(new CObject))
     {
      printf("inserisci errore");
      delete array;
      return;
     }
   //--- elimina array
   delete array;
  }
```

Ricerca

Cerca un elemento uguale al campione nell'array ordinato.

```
int  Search(
   CObject*  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayObj::Search(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- crea campione(sample)
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete array;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- cerca elemento
   if(array.Search(sample)!=-1) printf("Elemento trovato");
   else                         printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreat

Cerca un elemento con un valore superiore al valore del campione nell' array ordinato.

```
int  SearchGreat(
   CObject*  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayObj::SearchGreat(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- crea campione(sample)
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete array;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- cerca elemento
   if(array.SearchGreat(sample)!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLess

Ricerca un elemento con un valore inferiore rispetto al valore del campione nell'array ordinato.

```
int  SearchLess(
   CObject*  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayObj:: SearchLess(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- crea campione(sample)
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete array;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- cerca elemento
   if(array.SearchLess(sample)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchGreatOrEqual

La ricerca di un elemento con un valore maggiore o uguale al valore del campione nel vettore ordinato.

```
int  SearchGreatOrEqual(
   CObject*  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayObj::SearchGreatOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- crea campione(sample)
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete array;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- cerca elemento
   if(array.SearchGreatOrEqual(sample)!=-1) printf("Elemento trovato");
   else                                     printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLessOrEqual

Ricerca di un elemento con un valore inferiore o uguale al valore del campione nell'array ordinato.

```
int  SearchLessOrEqual(
   CObject*  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayObj:: SearchLessOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- crea campione(sample)
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete array;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- cerca elemento
   if(array.SearchLessOrEqual(sample)!=-1) printf("Elemento trovato");
   else                                    printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchFirst

Ricerca il primo elemento uguale al campione nell'array ordinato.

```
int  SearchFirst(
   CObject*  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayObj::SearchFirst(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- crea campione(sample)
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete array;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- cerca elemento
   if(array.SearchFirst(sample)!=-1) printf("Elemento trovato");
   else                              printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

SearchLast

Ricerca l'ultimo elemento uguale al campione nell'array ordinato.

```
int  SearchLast(
   CObject*  element      // campione(sample)
   ) const
```

Parametri

element

[in] L'elemento camptione da cercare nell'array.

Valore di ritorno

La posizione dell'elemento trovato - successo; invece -1 se l'elemento non è stato trovato.

Esempio:

```
//--- esempio per CArrayObj:: SearchLast(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- ordina elemento
   array.Sort();
   //--- crea campione(sample)
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete array;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- cerca elemento
   if(array.SearchLast(sample)!=-1) printf("Elemento trovato");
   else                             printf("Elemento non trovato");
   //--- elimina array
   delete array;
  }
```

Save

Salva l'array dati nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CArrayObj::Save(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiungi elementi dell'array
   //--- . . .
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

Load

Carica i dati array dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Nota

Durante la lettura elementi dell'array dal file, viene chiamato il metodo CArrayObj::CreateElement(int) per creare ogni elemento.

Esempio:

```
//--- esempio per CArrayObj::Load(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- errore caricamento file
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

```
   //--- usa elementi dell'array
   //--- . . .
   //--- elimina array
   delete array;
  }
```

Type

Ottiene l'identificatore tipo array.

```
virtual int  Type() const
```

Valore di ritorno

Tipo identificatore array (for CArrayObj - 7778).

Esempio:

```
//--- esempio per CArrayObj::Type()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo dell'array
   int type=array.Type();
   //--- elimina array
   delete array;
  }
```

CList

CList Class è una classe di lista dinamica di istanze della classe CObject e le sue classi derivate.

Descrizione

La Classe CList offre la possibilità di lavorare con un elenco di istanze di CObject e le sue classi derivate. La classe consente di aggiungere/inserire/cancellare elementi di un array, eseguire l'ordinamento di un array, e la ricerca in un array ordinato. Inoltre, sono stati implementati i metodi di lavoro con i file.

Ci sono alcune sottigliezze di lavoro con la classe CList. La classe ha un meccanismo di controllo della memoria dinamica, quindi state attenti quando lavorate con gli elementi della lista.

Sottigliezze del meccanismo di gestione della memoria simili a quelle descritte in CArrayObj.

Dichiarazione

```
class CList : public CObject
```

Titolo

```
#include <Arrays\List.mqh>
```

Gerarchia di ereditarietà

CObject
  CList

I Metodi della Classe per Gruppi

Attributi	
FreeMode	Ottiene la flag della gestione della memoria durante l'eliminazione di elementi della lista
FreeMode	Imposta la flag di gestione della memoria durante l'eliminazione di elementi della lista
Totale	Ottiene il numero di elementi nella lista
IsSorted	Ottiene la flag lista ordinata
SortMode	Ottiene la modalità di ordinamento
Metodi creazione	
CreateElement	Crea un nuovo elemento della lista
Aggiunta metodi	
Add	Aggiunge un elemento alla fine della lista
Insert	Inserisce un elemento alla posizione specificata della lista

Attributi	
Metodi eliminazione	
DetachCurrent	Rimuove un elemento dalla posizione corrente nella lista senza eliminarlo "fisicamente"
DeleteCurrent	Rimuove l'elemento dalla posizione corrente nella lista
Delete	Rimuove l'elemento dalla posizione specificata nella lista
Clear	Rimuove tutti gli elementi della lista
Navigazione	
IndexOf	Ottiene l'indice dell'elemento della lista specificata
GetNodeAtIndex	Ottiene un elemento con l'indice specificato della lista
GetFirstNode	Ottiene il primo elemento della lista
GetPrevNode	Ottiene l'elemento precedente della lista
GetCurrentNode	Ottiene l'elemento della lista corrente
GetNextNode	Ottiene l'elemento successivo nella lista
GetLastNode	Ottiene l'ultimo elemento della lista
Ordering methods	
Sort	Ordina la lista
MoveToIndex	Sposta l'elemento corrente nella lista alla posizione specificata
Exchange	Scambia due elementi nella lista
Confronta i metodi	
CompareList	Confronta la lista con un altra
Metodi di ricerca	
Ricerca	Cerca un elemento uguale al campione nella lista ordinata
Input/output	
virtual Save	Salva i dati della lista nel file
virtual Load	Carica i dati della lista dal file
virtual Type	Ottiene il identificatore tipo lista

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode

Ottiene la flag della gestione della memoria durante l'eliminazione di elementi della lista.

```
bool  FreeMode() const
```

Valore di ritorno

Flag della gestione della memoria.

Esempio:

```
//--- esempio per CList::FreeMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Erorre creazione oggetto");
      return;
     }
   //--- ottiene flag modalità free
   bool list_free_mode=list.FreeMode();
   //--- elimina lista
   delete list;
  }
```

FreeMode

Imposta il flag di gestione della memoria durante l'eliminazione di elementi della lista.

```
void  FreeMode(
   bool  mode      // nuovo valore
   )
```

Parametri

mode

[in] Nuovo valore del flag di gestione della memoria.

Nota

Impostare la flag di gestione della memoria è una parte importante della classe di utilizzo CList. Dal momento che gli elementi della lista sono puntatori ad oggetti dinamici, è importante per determinare che cosa fare con essi durante la rimozione dalla lista.

Se è impostato il flag, quando si rimuove un elemento dalla lista, l'elemento viene eliminato automaticamente dal l'operatore delete. Se il flag non è impostato, si presume che un puntatore all'oggetto eliminato è ancora da qualche parte nel programma utente e verrà deallocato dal programma più in la.

Se il programma utente ripristina la flag della gestione della memoria, gli utenti dovrebbero comprendere la loro responsabilità per la rimozione degli elementi della lista prima della terminazione del programma. Altrimenti, la memoria allocata per gli elementi del nuovo operatore non viene rilasciata.

Per grandi quantità di dati questo potrebbe portare anche ad un crash del terminale.

Se il programma utente non azzera il flag di gestione della memoria, c'è un altra "trappola"(pitfall). Quando gli elementi puntatore in una lista sono memorizzati da qualche parte nelle variabili locali, poi la rimozione della lista porterà ad un errore critico e crash del programma utente. Per default, viene impostato il flag di gestione della memoria, cioè la classe della lista è responsabile del rilascio degli elementi di memoria.

Esempio:

```
//--- esempio per CList::FreeMode(bool)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Erorre creazione oggetto");
      return;
     }
   //--- resetta modalità flag free
   list.FreeMode(false);
```

```
   //--- usa lista
   //--- . . .
   //--- elimina lista
   delete list;
  }
```

Totale

Ottiene il numero di elementi nella lista.

```
int  Total() const
```

Valore di ritorno

Numero di elementi nella lista.

Esempio:

```
//--- esempio per CList::Total()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controlla totale
   int total=list.Total();
   //--- usa lista
   //--- ...
   //--- elimina lista
   delete list;
  }
```

IsSorted

Ottiene la flag lista ordinata.

```
bool  IsSorted(
   int  mode=0      // modalità di ordinamento
   ) const
```

Parametri

mode=0

[in] Modalità di ordinamento controllata.

Valore di ritorno

Flag della lista ordinata. Restituisce true se l'elenco è ordinato utilizzando la modalità specificata, altrimenti restituisce false.

Nota

La flag della lista ordinata non può essere modificata direttamente. La flag è impostata per Sort(int) e si resetta da qualsiasi metodo add/insert.

Esempio:

```
//--- esempio per CList::IsSorted()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controlla l'ordinamento
   if(list.IsSorted(0))
     {
      //--- usa metodi per liste ordinate
      //--- ...
     }
   //--- elimina lista
   delete list;
  }
```

SortMode

Ottiene la versione di ordinamento.

```
int  SortMode() const
```

Valore di ritorno

Modalità di ordinamento, o -1 se la lista non è ordinata.

Esempio:

```
//--- esempio per CList::SortMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- controlla modalità ordinamento
   int sort_mode=list.SortMode();
   //--- usa lista
   //--- ...
   //--- elimina lista
   delete list;
  }
```

CreateElement

Crea un nuovo elemento della lista.

```
CObject*  CreateElement()
```

Valore di ritorno

Puntatore all'elemento appena creato - con successo, NULL - non si può creare un elemento.

Nota

Il Metodo CreateElement() nella classe CList restituisce sempre NULL e non esegue alcuna azione. Se necessario, il metodo CreateElement() dovrebbe essere attuato in una classe derivata.

Esempio:

```
//--- esempio per CList::CreateElement(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- riempie lista
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("Errore creazione elemento");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- usa lista
   //--- . . .
   //--- elimina lista
   delete list;
  }
```

Add

Aggiunge un elemento alla fine della lista.

```
int  Add(
   CObject*  element      // elemento da aggiungere
   )
```

Parametri

element

[in V valore dell'elemento da aggiungere alla lista.

Valore di ritorno

Indice dell'elemento aggiunto - successo, -1 - errore.

Nota

L'elemento non viene aggiunto all'elenco, se un puntatore non valido (per esempio, NULL) viene passato come parametro.

Esempio:

```
//--- esempio per CList::Add(Cobject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge 100 elementi
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("Errore aggiunta elemento");
         delete list;
         return;
        }
     }
   //--- usa lista
   //--- . . .
   //--- elimina lista
   delete list;
  }
```

Insert

Inserisce un elemento nella posizione specificata nella lista.

```
int  Insert(
   CObject*  element,     // elemento da inserire
   int       pos          // posizione
   )
```

Parametri

element

[in] valore dell'elemento da inserire nella lista

pos

[in] posizione nella lista dove inserire

Valore di ritorno

Indice dell' elemento inserito - successo, -1 - errore.

Nota

L'elemento non viene aggiunto all'elenco, se un puntatore non valido (per esempio, NULL) viene passato come parametro.

Esempio:

```
//--- esempio per CList::Insert(CObject*,int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- inserisce 100 elementi
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("Errore inserimento elemento");
         delete list;
         return;
        }
     }
   //--- usa lista
   //--- . . .
```

```
   //--- elimina lista
   delete list;
  }
```

DetachCurrent

Estrae un elemento dalla posizione corrente nella lista senza la sua eliminazione "fisica".

```
CObject*  DetachCurrent()
```

Valore di ritorno

Puntatore all'elemento rimosso in caso di successo, NULL - non è possibile rimuovere l'elemento.

Nota

Quando viene rimosso dalla lista, l'elemento non viene rimosso in qualsiasi stato di flag di gestione della memoria. Il puntatore all'elemento estratto deve essere rilasciato dopo che è stato usato.

Esempio:

```
//--- esempio per CList::DetachCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("Errore distaccamento");
      delete list;
      return;
     }
   //--- usa elemento
   //--- . . .
   //--- elimina elemento
   delete object;
   //--- elimina lista
   delete list;
  }
```

DeleteCurrent

Rimuove l'elemento dalla posizione corrente nella lista.

```
bool  DeleteCurrent()
```

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Nota

Se la gestione della memoria è attivata, la memoria per l'elemento rimosso viene deallocata.

Esempio:

```
//--- esempio per CList::DeleteCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   if(!list.DeleteCurrent())
     {
      printf("Errore eliminazione");
      delete list;
      return;
     }
   //--- elimina lista
   delete list;
  }
```

Delete

Rimuove l'elemento dalla posizione indicata nella lista.

```
bool  Delete(
   int  pos      // posizione
   )
```

Parametri

pos

[in] posizione dell'elemento da rimuovere dalla lista.

Valore di ritorno

true - successo, false - non si può rimuovere l'elemento.

Nota

Se è attivato il flag di gestione della memoria, la memoria utilizzata per l'elemento eliminato viene rilasciata.

Esempio:

```
//--- esempio per CList::Delete(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   if(!list.Delete(0))
     {
      printf("Errore eliminazione");
      delete list;
      return;
     }
   //--- elimina lista
   delete list;
  }
```

Clear

Rimuove tutti gli elementi della lista.

```
void  Clear()
```

Nota

Se la gestione della memoria è attivata, la memoria degli elementi eliminati viene deallocata.

Esempio:

```
//--- esempio per CList::Clear()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   //--- pulisce lista
   list.Clear();
   //--- elimina lista
   delete list;
  }
```

IndexOf

Ottiene l'indice dell'elemento dalla lista specificata.

```
int  IndexOf(
   CObject*  element      // puntatore all'elemento
   )
```

Parametri

element

[in] puntatore all'elemento della lista.

Valore di ritorno

Indice elemento lista, oppure -1.

Esempio:

```
//--- esempio per CList::IndexOf(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("Errore creazione elemento");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- elimina lista
   delete list;
  }
```

GetNodeAtIndex

Ottiene un elemento dalla posizione specificata nella lista.

```
CObject*  GetNodeAtIndex(
   int  pos      // posizione
   )
```

Parametri

pos

[in] posizione elemento nella lista.

Valore di ritorno

puntatore all'elemento - successo, NULL - non può ricevere un puntatore.

Esempio:

```
//--- example for CList::GetNodeAtIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Erorre creazione oggetto");
      return;
     }
   //--- aggiunge elementi alla lista
   //--- . . .
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("Ottiene errore del nodo");
      delete list;
      return;
     }
   //--- usa elemento
   //--- . . .
   //--- non eliminare elemento
   //--- elimina lista
   delete list;
  }
```

GetFirstNode

Ottiene il primo elemento della lista.

```
CObject*  GetFirstNode()
```

Valore di ritorno

Puntatore al primo elemento - successo, NULL - non si può ottenere un puntatore.

Esempio:

```
//--- esempio per CList::GetFirstNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("Ottiene errore nodo");
      delete list;
      return;
     }
   //--- usa elemento
   //--- . . .
   //--- non eliminare elemento
   //--- elimina lista
   delete list;
  }
```

GetPrevNode

Ottiene l'elemento precedente della lista.

```
CObject*  GetPrevNode()
```

Valore di ritorno

Puntatore all'elemento precedente - successo, NULL - non si può ottenere un puntatore.

Esempio:

```
//--- esempio per CList::GetPrevNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("Ottiene errore nodo");
      delete list;
      return;
     }
   //--- usa elemento
   //--- . . .
   //--- non eliminare elemento
   //--- elimina lista
   delete list;
  }
```

GetCurrentNode

Ottiene l'elemento della lista corrente.

```
CObject*  GetCurrentNode()
```

Valore di ritorno

Puntatore all'elemento corrente - successo, NULL - non si può ottenere un puntatore.

Esempio:

```
//--- esempio per CList::GetCurrentNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("Ottiene errore nodo");
      delete list;
      return;
     }
   //--- usa elemento
   //--- . . .
   //--- non eliminare elemento
   //--- elimina lista
   delete list;
  }
```

GetNextNode

Ottiene l'elemento successivo nella lista.

```
CObject*  GetNextNode()
```

Valore di ritorno

Puntatore all'elemento successivo - successo, NULL - non si può ottenere un puntatore.

Esempio:

```
//--- esempio per CList::GetNextNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("Ottiene errore nodo");
      delete list;
      return;
     }
   //--- usa elemento
   //--- . . .
   //--- non eliminare elemento
   //--- elimina lista
   delete list;
  }
```

GetLastNode

Ottiene l'ultimo elemento della lista.

```
CObject*  GetLastNode()
```

Valore di ritorno

Puntatore all'ultimo elemento - successo, NULL - non si può ottenere un puntatore.

Esempio:

```
//--- esempio per CList::GetLastNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi nella lista
   //--- . . .
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("Ottiene errore nodo");
      delete list;
      return;
     }
   //--- usa elemento
   //--- . . .
   //--- non eliminare elemento
   //--- elimina lista
   delete list;
  }
```

Sort

Ordina una lista.

```
void  Sort(
   int  mode      // modalità di ordinamento
   )
```

Parametri

mode

[in] Modalità ordinamento.

Valore di ritorno

No.

Nota

La lista è sempre ordinata in ordine crescente.

Esempio:

```
//--- esempio per CList::Sort(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ordina per modo 0
   list.Sort(0);
   //--- usa lista
   //--- ...
   //--- elimina lista
   delete list;
  }
```

MoveToIndex

Sposta l'elemento corrente nella lista alla posizione specificata.

```
bool  MoveToIndex(
   int  pos      // posizione
   )
```

Parametri

pos

[in] posizione nella lista in cui muoversi.

Valore di ritorno

vero - successo, false - non può spostare l'elemento.

Esempio:

```
//--- esempio per CList::MoveToIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- sposta il corrente nodo all'inizio
   list.MoveToIndex(0);
   //--- usa lista
   //--- . . .
   //--- elimina lista
   delete list;
  }
```

Exchange

Scambia due elementi nella lista.

```
bool  Exchange(
   CObject*  node1,     // lista elementi
   CObject*  node2      // lista elementi
   )
```

Parametri

node1

[in] list element

node2

[in] list element

Valore di ritorno

true - successo, false - non si possono scambiare gli elementi.

Esempio:

```
//--- esempio per CList::Exchange(CObject*,CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- lo scambio
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- usa lista
   //--- . . .
   //--- elimina lista
   delete list;
  }
```

CompareList

Confronta la lista con un altra.

```
bool  CompareList(
   CList*  list      // puntatore alla sorgente
   )
```

Parametri

list

[in] un puntatore ad un'istanza della classe CList in utilizzo come sorgente di elementi per il confronto.

Valore di ritorno

true - le liste sono uguali, false - le liste non sono uguali.

Esempio:

```
//--- esempio per CList::CompareList(const CList*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- crea lista sorgente
   CList *src=new CList;
   if(src==NULL)
     {
      printf("Errore creazione oggetto");
      delete list;
      return;
     }
   //--- riempie lista sorgente
   //--- . . .
   //--- confronta con un altra lista
   bool result=list.CompareList(src);
   //--- elimina liste
   delete src;
   delete list;
  }
```

Ricerca

Cerca un elemento uguale al campione nella lista ordinata.

```
CObject*  Search(
   CObject*  element      // campione(sample)
   )
```

Parametri

element

[in] campione elemento da cercare nella lista.

Valore di ritorno

Puntatore all'elemento trovato - successo, NULL - l'elemento non viene trovato.

Esempio:

```
//--- esempio per CList::Search(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi lista
   //--- . . .
   //--- lista ordinata
   list.Sort(0);
   //--- crea campione
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Errore creazione campione");
      delete list;
      return;
     }
   //--- imposta attributi campione
   //--- . . .
   //--- imposta elementi
   if(list.Search(sample)!=NULL) printf("Elemento trovato");
   else                          printf("Elemento trovato");
   //--- elimina lista
   delete list;
  }
```

Save

Salva i dati della lista nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen().

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CList::Save(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- aggiunge elementi lista
   //--- . . .
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("File save: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- elimina lista
   delete list;
```

```
}
```

Load

Carica lista dati dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario precedentemente aperto con la funzione FileOpen().

Valore di ritorno

true - completato con successo, false - errore.

Nota

Durante la lettura elementi della lista dal file, il metodo CList::CreateElement(int) viene chiamato per creare ogni elemento.

Esempio:

```
//--- esempio per CLoad::Load(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- errore caricamento file
         printf("Caricamento file: Errore %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

```
   //--- usa elementi della lista
   //--- . . .
   //--- elimina lista
   delete list;
  }
```

Type

Ottiene l'identificatore del tipo di elenco.

```
virtual int  Type()
```

Valore di ritorno

Identificatore tipo Lista(per CList - 7779).

Esempio:

```
//--- esempio per CList::Type()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Errore creazione oggetto");
      return;
     }
   //--- ottiene il tipo di lista
   int type=list.Type();
   //--- elimina lista
   delete list;
  }
```

CTreeNode

La classe CTreeNode è una classe del nodo albero binario Ctree.

Descrizione

CTreeNode offre la possibilità di lavorare con i nodi dell'albero binario Ctree. Le opzioni di navigazione attraverso l'albero sono implementate nella classe. Inoltre, sono implementati i metodi di lavoro con il file.

Dichiarazione

```
class CTreeNode : public CObject
```

Titolo

```
#include <Arrays\TreeNode.mqh>
```

Gerarchia di ereditarietà

CObject

CTreeNode

Discendenti diretti

CTree

I Metodi della Classe per Gruppi

Attributi	
Owner	Ottiene/imposta il puntatore del nodo proprietario
Left	Ottiene/imposta il puntatore del nodo di sinistra
Right	Ottiene/imposta il puntatore del nodo di destra
Balance	Ottiene il bilanciamento del nodo
BalanceL	Ottiene il bilancio del sotto-ramo sinistro del nodo
BalanceR	Ottiene il bilancio del sotto-ramo destro del nodo
Creazione di un nuovo elemento	
CreateSample	Crea una nuova istanza del nodo
Confronto	
RefreshBalance	Ricalcola il bilanciamento del nodo
Ricerca	
GetNext	Ottiene il puntatore del nodo successivo
Input/Output	

Attributi	
SaveNode	Salva i dati del nodo in un file
LoadNode	Scarica i dati del nodo da un file
virtual Type	Ottiene l'identificatore del tipo di nodo

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Compare

Alberi di classi discendenti di CTreeNode hanno applicazione pratica.

Un discendente della classe CTreeNode dovrebbe avere metodi predefiniti: CreateSample crea una nuova istanza della classe discendente di CTreeNode, Compare confronta i valori di campi chiave della classe discendente di CTreeNode, Type (se è necessario identificare un nodo), SaveNode e LoadNode (se è necessario lavorare con un file).

Prendiamo in considerazione un esempio di una classe discendente Ctree.

```
//+------------------------------------------------------------------+
//|                                                   MyTreeNode.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                             https://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+------------------------------------------------------------------+
//| Descrivi la classe CMyTreeNode derivata da CTreeNode.               |
//+------------------------------------------------------------------+
//| Classe CMyTreeNode.                                                |
//| Scopo: Classe di elemento di un albero binario.                      |
//|              Descendente della classe CTreeNode.                      |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- dati utente
   long              m_long;           // campo chiave di tipo long
   double            m_double;         // variabile personalizzata di tipo double
   string            m_string;         // variable personalizzata di tipo string
   datetime          m_datetime;       // variable personalizzata di tipo datetime

public:
                     CMyTreeNode();
   //--- metodi di accesso ai dati utente
   long              GetLong(void)                { return(m_long); }
   void              SetLong(long value)          { m_long=value;   }
```

```
   double            GetDouble(void)               { return(m_double); }
   void              SetDouble(double value)       { m_double=value;  }
   string            GetString(void)               { return(m_string); }
   void              SetString(string value)       { m_string=value;  }
   datetime          GetDateTime(void)             { return(m_datetime); }
   void              SetDateTime(datetime value)   { m_datetime=value;  }
   //--- metodi per il lavoro con i file
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- metodi di creazione di istanze della classe
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| CMyTreeNode costruttore della classe.                            |
//| INPUT:  niente.                                                  |
//| OUTPUT: niete.                                                   |
//| REMARK: niente.                                                  |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- inizializzazione dei dati utente
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| Confronto con un altro nodo dell'albero per algoritmo specificato.|
//| INPUT:  nodo - elementi dell'albero da comparare,                |
//|         mode - identificatore dell'algoritmo di confronto.       |
//| OUTPUT: risultato del confronto (>0,0,<0).                       |
//| REMARK: niente.                                                  |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- il parametro mode viene ignorato, perché l'algoritmo di costruzione albero è l'u
   int res=0;
//--- type casting esplicito
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Creazione di una nuova istanza della classe.                     |
//| INPUT:  niente.                                                  |
//| OUTPUT: puntatore ad una nuova istanza della classe CMyTreeNode. |
```

```
//| REMARK: niente.                                                  |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| Scrive i dati nodo dell'albero in un file.                       |
//| INPUT:  file_handle -handle di un file pre-aperto per la scrittura.    |
//| OUTPUT: true se OK, altrimenti false.                            |
//| REMARK: niente.                                                  |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- controllo
   if(file_handle<0) return(false);
//--- scrive dati utente
//--- scrittura variabile personalizzata di tipo long
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))          return(false);
//--- scrittura variabile personalizzata di tipo double
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))    return(false);
//--- scrittura variabile personalizzata di tipo string
   len=StringLen(m_string);
//--- scrive la lunghezza della stringa
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)   return(false);
//--- scrive la stringa
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- scrittura variabile personalizzata di tipo datetime
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))      return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Legge i dati nodo dell'albero da un file.                        |
//| INPUT:  file_handle -handle di un file pre-aperto per la lettura.    |
//| OUTPUT: true se OK, altrimenti false.                            |
//| REMARK: niente.                                                  |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- controllo
   if(file_handle<0) return(false);
//--- lettura
   if(FileIsEnding(file_handle)) return(false);
//--- lettura variabile personalizzata di tipo char
```

```
//--- lettura variabile personalizzata di tipo long
   m_long=FileReadLong(file_handle);
//--- lettura variabile personalizzata di tipo double
   m_double=FileReadDouble(file_handle);
//--- lettura variabile personalizzata di tipo string
//--- legge la lunghezza della stringa
   len=FileReadInteger(file_handle,INT_VALUE);
//--- legge la stringa
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- lettura variabile personalizzata di tipo datetime
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

Owner

Ottiene il puntatore del nodo proprietario.

```
CTreeNode*  Owner()
```

Valore di ritorno

Puntatore del nodo del proprietario.

Owner

Imposta il puntatore del nodo proprietario.

```
void  Owner(
   CTreeNode*  node      // nodo
   )
```

Parametri

node

[in] Nuovo valore del puntatore del nodo del proprietario.

Valore di ritorno

Nessuno.

Left

Ottiene il puntatore del nodo di sinistra.

```
CTreeNode*  Left()
```

Valore di ritorno

Puntatore del nodo di sinistra.

Left

Imposta il puntatore del nodo di sinistra.

```
void  Left(
   CTreeNode*  node       // nodo
   )
```

Parametri

node

[in] Nuovo valore del puntatore del nodo di sinistra.

Valore di ritorno

Nessuno.

Right

Ottiene il puntatore del nodo di destra

```
CTreeNode*  Right()
```

Valore di ritorno

Il puntatore del nodo di destra.

Right

Imposta il puntatore del nodo di destra.

```
void  Right(
   CTreeNode*  node      // nodo
   )
```

Parametri

node

[in] Nuovo valore del puntatore del nodo di destra.

Valore di ritorno

Nessuno.

Balance

Ottiene il bilanciamento nodo.

```
int  Balance() const
```

Valore di ritorno

Bilanciamento nodo.

BalanceL

Ottiene il bilanciamento del sotto-ramo di sinistra del nodo.

```
int  BalanceL() const
```

Valore di ritorno

Bilanciamento del sub-ramo sinistro del nodo.

BalanceR

Ottiene il bilanciamento del sotto-ramo destro del nodo

```
int  BalanceR() const
```

Valore di ritorno

Bilanciamento del sub-ramo destro del nodo.

CreateSample

Crea un nuovo campione nodo.

```
virtual CTreeNode*  CreateSample()
```

Valore di ritorno

Puntatore al nuovo campione nodo o NULL.

RefreshBalance

Ricalcola il bilanciamento del nodo.

```
int  RefreshBalance()
```

Valore di ritorno

Bilanciamento nodo.

GetNext

Ottiene il puntatore del nodo successivo.

```
CTreeNode*  GetNext(
   CTreeNode*  node       // nodo
   )
```

Parametri

node

[in] Nodo dell'inizio ricerca.

Valore di ritorno

Puntatore del nodo successivo.

SaveNode

Scrive i dati del nodo in un file.

```
bool  SaveNode(
   int  file_handle      // handle
   )
```

Parametri

file_handle

[in] Handle di un file binario che è stato in precedenza aperto per la scrittura.

Valore di ritorno

true - successo, altrimenti false.

LoadNode

Legge i dati dei nodi da un file.

```
bool  LoadNode(
   int          file_handle,      // handle
   CTreeNode*   main              // nodo
   )
```

Parametri

file_handle

[in] Handle di un file binario che è stato in precedenza aperto in lettura.

main

[in] Nodo per i dati.

Valore di ritorno

true - successo, altrimenti false.

Type

Ottiene l'identificatore del tipo di nodo.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo di nodo.

CTree

CTree è una classe di albero binario delle istanze della classe CTreeNode e dei suoi discendenti.

Descrizione

La Classe Ctree offre la possibilità di lavorare con l'albero binario di istanze della classe CTreeNode e dei suoi discendenti. Opzioni di aggiunta/inserimento/cancellazione di elementi albero e ricerca nell' albero vengono implementati nella classe. Oltre a questo, vengono implementati i metodi di lavoro con un file.

Si noti che il meccanismo di gestione della memoria dinamica non è implementato nella classe Ctree (a differenza della classi CList e CArrayObj). Tutti i nodi dell'albero vengono cancellati con la deallocazione di memoria.

Dichiarazione

```
class CTree : public CTreeNode
```

Titolo

```
#include <Arrays\Tree.mqh>
```

Gerarchia di ereditarietà

CObject
  CTreeNode
    CTree

I Metodi della Classe per Gruppi

Attributi	
Root	Ottiene il nodo radice(root) dell'albero
Creazione di un nuovo elemento	
CreateElement	Crea una nuova istanza del nodo
Filling	
Insert	Aggiunge un nodo ad un albero
Eliminazione	
Detach	Distacca un nodo specificato da un albero
Delete	Elimina un nodo specificato da un albero
Clear	Elimina tutti i nodi di un albero
Ricerca	
Find	Ricerche per un nodo in un albero per campione

Attributi	
Input/output	
virtual Save	Salva tutti i dati di un albero in un file
virtual Load	Scarica i dati di un albero da un file
virtual Type	Ottiene identificativo del tipo di albero

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CTreeNode

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

Alberi dei discendenti della classe CTreeNode - discendenti della classe Ctree - hanno applicazione pratica.

Discendente della classe Ctree dovrebbero avere un metodo predefinito CreateElement che crea una nuova istanza della classe discendente CTreeNode.

Prendiamo in considerazione un esempio della classe discendente Ctree.

```
//+------------------------------------------------------------------+
//|                                                      MyTree.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//| www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+------------------------------------------------------------------+
//| Descrive la classe CMyTree derivata da CTree.                     |
//+------------------------------------------------------------------+
//| Classe CMyTree.                                                  |
//| Scopo: Costruzione e navigazione di una ricerca albero binario.    |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- medoti di ricerca nell'albero da dati personalizzati
   CMyTreeNode*        FindByLong(long find_long);
   //--- metodo di creazione di elementi dell'albero
   virtual CTreeNode *CreateElement();
  };
```

```
//---
CMyTree MyTree;
//+------------------------------------------------------------------+
//| Creazione di un nuovo nodo albero.                               |
//| INPUT:  niente.                                                  |
//| OUTPUT: puntatore al nuovo nodo albero se OK, o NULL.            |
//| REMARK: niente.                                                  |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//| Ricerca di elemento in una lista per valore m_long.              |
//| INPUT:  find_long - valore cercato.                              |
//| OUTPUT: puntatore all'elemento della lista trovato, oppure NULL. |
//| REMARK: niente.                                                  |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- crea un nodo della struttura per passare il parametro di ricerca
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| script "testing della classe CMyTree"                            |
//+------------------------------------------------------------------+
//---  array per l'inizializzazione stringa
string str_array[11]={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","999
//---
int OnStart() export
  {
   int          i;
   uint         pos;
   int          beg_time,end_time;
   CMyTreeNode *node; //--- puntatore temporaneo al campione della classe CMyTreeNode
//---
   printf("Inizio test %s.",__FILE__);
// --- Compila MyTree con istanze della classe MyTreeNode nella quantità di extCounte
```

```
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i++)
     {
      node=MyTree.CreateElement();
      if(node==NULL)
        {
         //--- uscita di emergenza
         printf("%s (%4d): crea errore",__FILE__,__LINE__);
         return(__LINE__);
        }
      NodeSetData(node,i);
      node.SetLong(i);
      MyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Tempo di ricerca di MyTree è di %d ms.",end_time-beg_time);
//--- Crea un albero TmpMyTree temporaneo.
   CMyTree TmpMyTree;
//--- Distacca il 50% degli elementi di albero (tutti pari)
//--- e li aggiunger allalbero temporaneo TmpMyTree.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i+=2)
     {
      node=MyTree.FindByLong(i);
      if(node!=NULL)
         if(MyTree.Detach(node)) TmpMyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Tempo di eliminazione di %d elementi da MyTree è di %d ms.",extCountedNodes
//--- Restituisce il distaccamento
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Controlla il lavoro del metodo Save(int file_handle);
   int file_handle;
   file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- errore di scrittura in un file
         //--- uscita di emergenza
         printf("%s: Errore %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- chiusura file prima di lasciarlo!!!
         FileClose(file_handle);
         return(__LINE__);
```

```
        }
      FileClose(file_handle);
     }
//--- Controlla il lavoro del metodo Load(int file_handle);
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- errore di lettura dal file
         //--- uscita di emergenza
         printf("%s: Errore %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- chiusura file prima di lasciarlo!!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("Fine test %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Funzione per dare in output il contenuto nodo nel journal |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
              node.GetLong(),node.GetDouble(),
              node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| Funzione per popolare il nodo con valori casuali |
//+------------------------------------------------------------------+
void NodeSetData(CMyTreeNode *node,int mode)
  {
   if(mode%2==0)
     {
      node.SetLong(mode*MathRand());
      node.SetDouble(MathPow(2.02,mode)*MathRand());
     }
   else
     {
      node.SetLong(mode*(long)(-1)*MathRand());
      node.SetDouble(-MathPow(2.02,mode)*MathRand());
```

```
      }
    node.SetString(str_array[mode%10]);
    node.SetDateTime(10000*mode);
   }
```

Root

Ottiene il nodo radice dell'albero.

```
CTreeNode*  Root() const
```

Valore di ritorno

Puntatore del nodo radice dell'albero.

CreateElement

Crea una nuova istanza del nodo.

```
virtual CTreeNode*  CreateElement()
```

Valore di ritorno

Puntatore della nuova istanza del nodo o NULL.

Insert

Aggiunge un nodo ad un albero.

```
CTreeNode*  Insert(
   CTreeNode*  new_node      // nodo
   )
```

Parametri

new_node

[in] Puntatore di un nodo da inserire ad un albero.

Valore di ritorno

Puntatore del proprietario del nodo o NULL.

Detach

Stacca un nodo specificato da un albero.

```
bool  Detach(
   CTreeNode*  node       // nodo
   )
```

Parametri

node

[in] Puntatore nodo da staccare.

Valore di ritorno

true - successo, altrimenti false.

Nota

Dopo il distacco, il puntatore nodo non viene rilasciato. L'albero è bilanciato.

Delete

Elimina un nodo specificato da un albero.

```
bool  Delete(
   CTreeNode*  node      // nodo
   )
```

Parametri

node

[in] Puntatore nodo da eliminare.

Valore di ritorno

true - successo, altrimenti false.

Nota

Dopo la cancellazione, viene rilasciato un puntatore nodo. L'albero è bilanciato.

Clear

Elimina tutti i nodi di un albero.

```
void  Clear()
```

Valore di ritorno

Nessuno.

Nota

Dopo la cancellazione, i puntatori dei nodi vengono rilasciati.

Find

Cerca un un nodo in un albero per campione.

```
CTreeNode*  Find(
   CTreeNode*  node       // nodo
   )
```

Parametri

node

[in] Nodo che contiene i dati utilizzati come campione di ricerca.

Valore di ritorno

Puntatore del nodo trovato o NULL.

Save

Scrive i dati albero in un file.

```
virtual bool  Save(
   int  file_handle      // handle
   )
```

Parametri

file_handle

[in] Handle di un file binario che è stato in precedenza aperto per la scrittura.

Valore di ritorno

true - successo, altrimenti false.

Load

Legge i dati albero da un file.

```
virtual bool  Load(
   int  file_handle      // handle
   )
```

Parametri

file_handle

[in] Handle di un file binario che è stato in precedenza aperto in lettura.

Valore di ritorno

true - successo, altrimenti false.

Type

Ottiene identificativo del tipo di albero

```
virtual int  Type() const
```

Valore di ritorno

Identificativo del tipo di albero

Collezioni Dati Generali

La libreria fornisce classi ed interfacce che definiscono le collezioni generiche, che consentono agli utenti di creare raccolte fortemente tipizzate. Queste collezioni offrono maggiore convenienza e performance nella manipolazione dei dati rispetto alle raccolte tipizzate non-generiche.

La libreria è disponibile nella cartella Include\Generic della directory di lavoro del terminale.

Oggetti:

Oggetto	Descrizione	Tipo
ICollection	Interfaccia per l'implementazione di raccolte di dati generiche	INTERFACE
IEqualityComparable	Interfaccia per l'implementazione di oggetti che possono essere confrontati	INTERFACE
IComparable	Interfaccia per l'implementazione di oggetti che possono essere confrontati in termini di "maggiore, minore o uguale a"	INTERFACE
IComparer	Interfaccia per l'implementazione di una classe generica che confronta due oggetti del tipo T, se uno è "maggiore di, minore di, o uguale all" altro	INTERFACE
IEqualityComparer	Interfaccia per l'implementazione di una classe generica che confronta due oggetti del tipo T per uguaglianza	INTERFACE
IList	Interfaccia per l'implementazione di elenchi di dati generici	INTERFACE
IMap	Interfaccia per l'implementazione di generiche collezioni di coppie chiave/valore	INTERFACE
ISet	Interfaccia per l'implementazione di set di dati generici	INTERFACE
CDefaultComparer	Una classe di assistente che implementa l' interfaccia generica IComparer<T> basata sul Confrontare(Compare) i metodi globali	CLASS
CDefaultEqualityComparer	Una classe helper che implementa l'interfaccia generica IEqualityComparer<T> utilizzando i metodi globali Equals<T> e GetHashCode	CLASS

Oggetto	Descrizione	Tipo
CArrayList	Una classe generica che implementa l'interfaccia IList <T>	CLASS
CKeyValuePair	La classe implementa la coppia chiave/valore	CLASS
CHashMap	Una classe generica che implementa l'interfaccia IMap <TKey, TValue>	CLASS
CHashSet	Una classe generica che implementa l'interfaccia ISet <T>	CLASS
CLinkedListNode	Una classe helper per implementare la classe CLinkedListNode<T>	CLASS
CLinkedList	Una classe generica che implementa l'interfaccia ICollection<T>	CLASS
CQueue	Una classe generica che implementa l'interfaccia ICollection<T>	CLASS
CRedBlackTreeNode	Una classe helper utilizzata per implementare la classe CRedBlackTree<T>	CLASS
CRedBlackTree	Una classe generica che implementa l'interfaccia ICollection<T>	CLASS
CSortedMap	Una classe generica che implementa l'interfaccia IMap <TKey, TValue>	CLASS
CSortedSet	Una classe generica che implementa l'interfaccia ISet <T>	CLASS
CStack	Una classe generica che implementa l'interfaccia ICollection<T>	CLASS

Metodi globali:

Metodo	Descrizione
ArrayBinarySearch	Cerca il valore specificato in un array unidimensionale ordinato in ordine crescente utilizzando l'interfaccia IComparable<T> per confrontare gli elementi
ArrayIndexOf	Cerca la prima occorrenza di un valore in un array unidimensionale
ArrayLastIndexOf	Cerca l'ultima occorrenza di un valore in un array unidimensionale
ArrayReverse	Modifica la sequenza di elementi in un array unidimensionale
Compare	Confronta due valori, se uno di essi è maggiore dell, inferiore dell, o uguale all, altro

Metodo	Descrizione
Equals	Confronta due valori per uguaglianza
GetHashCode	Calcola il valore del codice hash

ICollection<T>

ICollection <T> è un'interfaccia per l'implementazione di generiche raccolte di dati.

Descrizione

L'interfaccia ICollection<T> determina i metodi di base per lavorare con le raccolte, inclusi i metodi per conteggiare elementi, per cancellare una raccolta, aggiungere o eliminare elementi, ed altro.

Dichiarazione

```
template<typename T>
interface ICollection
```

Header

```
#include <Generic\Interfaces\ICollection.mqh>
```

Inheritance Hierarchy

ICollection

Direct descendants

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

Class Methods

Metodo	Descrizione
Add	Aggiunge un elemento ad una raccolta
Count	Restituisce il numero di elementi di una raccolta
Contains	Determina se una raccolta contiene un elemento con il valore specificato
CopyTo	Copia tutti gli elementi di una raccolta nell'array specificato partendo dall'indice specificato
Clear	Rimuove tutti gli elementi da una raccolta
Remove	Rimuove la prima occorrenza dell'elemento specificato da una raccolta

Add

Aggiunge un elemento ad una raccolta.

```
bool Add(
   T  value     // il valore dell'elemento
   );
```

Parametri

value

[in] Il valore dell'elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi di una raccolta.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Contains

Determina se una raccolta contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nella raccolta, altrimenti false.

CopyTo

Copia tutti gli elementi di una raccolta nell'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // indice iniziale per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritti gli elementi della raccolta.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

Clear

Rimuove tutti gli elementi di una raccolta.

```
void Clear();
```

Remove

Rimuove la prima occorrenza dell'elemento specificato da una raccolta.

```
bool Remove(
   T  item     // il valore dell'elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

IEqualityComparable<T>

IEqualityComparable<T> è un'interfaccia per implementare oggetti che possono essere confrontati.

Descrizione

L'interfaccia IEqualityComparable<T> definisce i metodi per recuperare il codice hash dell'oggetto corrente e verificare se è uguale ad un altro oggetto dello stesso tipo.

Dichiarazione

```
template<typename T>
interface IEqualityComparable
```

Header

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

Inheritance Hierarchy

IEqualityComparable

Direct descendants

IComparable

Class Methods

Metodo	Descrizione
Equals	Confronta l'oggetto corrente con il valore specificato
HashCode	Calcola il valore del codice hash per l'oggetto corrente

Equals

Confronta l'oggetto corrente con il valore specificato.

```
bool Equals(
   T  value     // il valore da confrontare
   );
```

Parametri

value

[in] Il valore con cui confrontare l'oggetto corrente.

Valore di ritorno

Restituisce true se gli oggetti sono uguali, altrimenti false.

HashCode

Calcola il valore del codice hash per l'oggetto corrente.

```
int HashCode();
```

Valore di ritorno

Restituisce il codice hash.

IComparable<T>

IComparable <T> è un'interfaccia per implementare oggetti che possono essere confrontati per scoprire se uno è maggiore di, inferiore o uguale ad, un altro

Descrizione

L'interfaccia IComparable<T> definisce un metodo per confrontare l'oggetto corrente con un altro oggetto dello stesso tipo, in base alla quale la raccolta di questi oggetti può essere ordinata.

Dichiarazione

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

Header

```
#include <Generic\Interfaces\IComparable.mqh>
```

Inheritance Hierarchy

IEqualityComparable

IComparable

Direct descendants

CKeyValuePair

Class Methods

Metodo	Descrizione
Compare	Confronta l'oggetto corrente con il valore specificato

Compare

Confronta l'oggetto corrente con il valore specificato.

```
int Compare(
   T  value     // il valore da confrontare
   );
```

Parametri

value

[in] Il valore con cui confrontare l'oggetto corrente.

Valore di ritorno

Restituisce un numero che esprime il rapporto dell'oggetto corrente e passato:

- se il risultato è inferiore a zero, l'oggetto corrente è inferiore a quello passato
- se il risultato è zero, l'oggetto corrente è uguale a quello passato
- se il risultato è maggiore di zero, l'oggetto corrente è maggiore di quello passato

IComparer<T>

IComparer <T> è un'interfaccia per l'implementazione di una classe generica che confronta due oggetti di tipo T, se uno è maggiore di, inferiore o uguale ad, un altro

Descrizione

L'interfaccia IComparer<T> determina un metodo per confrontare due oggetti di tipo T, sulla base della quale una raccolta di questi oggetti può essere ordinata.

Dichiarazione

```
template<typename T>
interface IComparer
```

Header

```
#include <Generic\Interfaces\IComparer.mqh>
```

Inheritance Hierarchy

IComparer

Direct descendants

CDefaultComparer

Class Methods

Metodo	Descrizione
Compare	Confronta due valori di tipo T

Compare

Confronta due valori di tipo T.

```
int Compare(
   T  x,      // il primo valore
   T  y       // il secondo valore
   );
```

Parametri

x

[in] Il primo valore da confrontare.

y

[in] Il primo valore da confrontare.

Valore di ritorno

Restituisce un numero che esprime il rapporto tra i due valori confrontati:

- se il risultato è inferiore a zero, x è inferiore a y (x<y)
- se il risultato è uguale a zero, x è uguale a y (x=y)
- se il risultato è maggiore di zero, x è maggiore di y (x>y)

IEqualityComparer<T>

IEqualityComparer<T> è un'interfaccia per l'implementazione di una classe generica che confronta due oggetti del tipo T.

Descrizione

L'interfaccia IEqualityComparer<T> definisce i metodi per recuperare il codice hash da un oggetto di tipo T e per verificare se due oggetti di tipo T sono uguali.

Dichiarazione

```
template<typename T>
interface IEqualityComparer
```

Header

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

Inheritance Hierarchy

IEqualityComparer

Direct descendants

CDefaultEqualityComparer

Class Methods

Metodo	Descrizione
Equals	Confronta due valori di tipo T
HashCode	Calcola il valore del codice hash basato sull'oggetto di tipo T

Equals

Confronta due valori di tipo T.

```
bool Equals(
   T  x,      // il primo valore
   T  y       // il secondo valore
   );
```

Parametri

x

[in] Il primo valore da confrontare.

y

[in] Il secondo valore da confrontare.

Valore di ritorno

Restituisce true se i valori sono uguali, altrimenti false.

HashCode

Calcola il valore del codice hash basato sull'oggetto tipo T.

```
int HashCode(
   T  value     // un oggetto per il calcolo
   );
```

Parametri

value

[in] L'oggetto per cui si desidera ottenere il codice hash.

Valore di ritorno

Restituisce il codice hash.

IList<T>

IList <T> è un'interfaccia per l'implementazione di elenchi di dati generici.

Descrizione

L'interfaccia IList<T> definisce i metodi di base per funzionare con le liste, come l'accesso ad un elemento per indice, per la ricerca e l'eliminazione di elementi, ordinamento, ed altro.

Dichiarazione

```
template<typename T>
interface IList : public ICollection<T>
```

Header

```
#include <Generic\Interfaces\IList.mqh>
```

Inheritance Hierarchy

ICollection

IList

Direct descendants

CArrayList

Class Methods

Metodo	Descrizione
TryGetValue	Ottiene un elemento dalla lista nell'indice specificato
TrySetValue	Modifica un valore dalla lista all'indice specificato
Insert	Inserisce un elemento nell'elenco nell'indice specificato
IndexOf	Cerca la prima occorrenza di un valore in un elenco
LastIndexOf	Cerca l'ultima occorrenza di un valore in un elenco
RemoveAt	Elimina un elemento dalla lista nell'indice specificato

TryGetValue

Ottiene un elemento della lista nell'indice specificato.

```
bool TryGetValue(
   const int  index,      // indice elemento
   T&         value       // la variabile per la scrittura
   );
```

Parametri

index

[in] L'indice dell'elemento dalla lista.

&value

[out] La variabile in cui verrà scritto il valore specificato dell'elemento dalla lista.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TrySetValue

Modifica un valore dalla lista all'indice specificato.

```
bool TrySetValue(
   const int  index,      // indice elemento
   T          value       // nuovo valore
   );
```

Parametri

index

[in] L'indice dell'elemento dalla lista.

value

[in] Il nuovo valore da assegnare all'elemento di lista specificato.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Insert

Inserisce un elemento nell'elenco nell'indice specificato.

```
bool Insert(
   const int  index,      // indice di dove inserire
   T          item        // il valore da inserire
   );
```

Parametri

index

[in] L'indice in cui inserire.

item

[in] Il valore da inserire nell'indice specificato.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

IndexOf

Cerca la prima occorrenza di un valore in un elenco.

```
int IndexOf(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce l'indice del primo elemento trovato. Se il valore non viene trovato, restituisce -1.

LastIndexOf

Cerca l'ultima occorrenza di un valore nell'elenco.

```
int LastIndexOf(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce l'indice dell'ultimo elemento trovato. Se il valore non viene trovato, restituisce -1.

RemoveAt

Elimina un elemento dalla lista nell'indice specificato.

```
bool RemoveAt(
   const int  index     // indice elemento
   );
```

Parametri

index

[in] L'indice dell'elemento che si desidera eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

IMap<TKey, TValue>

iMap<TKey, TValue> è un'interfaccia per implementare generiche raccolte di coppie chiave/valore.

Descrizione

L'interfaccia IMap<TKey, TValue> definisce i metodi di base per lavorare con le raccolte i cui dati sono memorizzati come coppie chiave/valore.

Dichiarazione

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

Header

```
#include <Generic\Interfaces\IMap.mqh>
```

Inheritance Hierarchy

ICollection

IMap

Direct descendants

CHashMap, CSortedMap

Class Methods

Metodo	Descrizione
Add	Aggiunge una coppia di chiave/valore ad una raccolta
Contains	Determina se una raccolta contiene la tabella chiave/valore con la chiave specificata
Remove	Rimuove la prima occorrenza di una coppia di chiave/valore da una raccolta
TryGetValue	Ottiene un elemento con la chiave specificata da una raccolta
TrySetValue	Modifica il valore della coppia di chiave/valore da una raccolta alla chiave specificata
CopyTo	Copia tutte le coppie chiave/valore di una raccolta in array specificati, a partire dall'indice specificato

Add

Aggiunge una coppia chiave/valore ad una raccolta.

```
bool Add(
   TKey    key,       // chiave
   TValue  value      // valore
   );
```

Parametri

key

[in] Chiave.

value

[in] Valore.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Contains

Determina se una raccolta contiene la tabella chiave/valore con la chiave specificata.

```
bool Contains(
   TKey    key,       // key
   TValue  value      // valore
   );
```

Parametri

key

[in] Chiave.

value

[in] Valore.

Valore di ritorno

Restituisce true, se la raccolta contiene la coppia di chiave/valore con la chiave e il valore specificati, altrimenti false.

Remove

Rimuove la prima occorrenza di una coppia di chiave/valore da una raccolta.

```
bool Remove(
   TKey  key     // chiave
   );
```

Parametri

key

[in] Chiave.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TryGetValue

Ottiene un elemento con la chiave specificata da una raccolta.

```
bool TryGetValue(
   TKey      key,       // chiave
   TValue&   value      // la variabile per scrivere i valori
   );
```

Parametri

key

[in] Chiave.

&value

[out] La variabile a cui verrà scritto il valore specificato della coppia chiave/valore.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TrySetValue

Modifica il valore della coppia chiave/valore dalla raccolta alla chiave specificata.

```
bool TrySetValue(
   TKey    key,       // chiave
   TValue  value      // nuovo valore
   );
```

Parametri

key

[in] Chiave.

value

[in] Il nuovo valore da assegnare alla coppia di chiave-valore specificata.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CopyTo

Copia tutte le coppie chiave/valore da una raccolta all'array specificato, a partire dall'indice specificato.

```
int CopyTo(
   TKey&       dst_keys[],       // un array per scrivere le chiavi
   TValue&     dst_values[],     // un array per scrivere i valori
   const int   dst_start=0       // l'indice iniziale per la scrittura
   );
```

Parametri

&dst_keys[]

[out] Un array nel quale verranno scritte tutte le chiavi della raccolta.

&dst_values[]

[out] Un array nel quale verranno scritti i valori delle corrispondenti chiavi, della raccolta.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di coppie di chiave/valore copiate.

ISet<T>

ISet<T> è un'interfaccia per l'implementazione di set di dati generici.

Descrizione

L'interfaccia ISet definisce i metodi di base per lavorare con i set, quali: l'unione e l'intersezione di set, la definizione di subsets stretti e non-stretti, ed altro.

Dichiarazione

```
template<typename T>
interface ISet : public ICollection<T>
```

Header

```
#include <Generic\Interfaces\ISet.mqh>
```

Inheritance Hierarchy

ICollection

ISet

Direct descendants

CHashSet, CSortedSet

Class Methods

Metodo	Descrizione
ExceptWith	Produce l'operazione di differenza tra la raccolta corrente e una raccolta passata (array)
IntersectWith	Produce il funzionamento dell'intersezione della raccolta corrente ed una raccolta passata (array)
SymmetricExceptWith	Produce l'operazione di differenza simmetrica tra la raccolta corrente ed una raccolta passata (array)
UnionWith	Produce l'unione della raccolta corrente ed una raccolta passata (array)
IsProperSubsetOf	Determina se il set corrente è un sottoinsieme appropriato della raccolta o dell'array specificati
IsProperSupersetOf	Determina se il set corrente è un superset appropriato della raccolta o array specificati
IsSubsetOf	Determina se il set corrente è un subset della raccolta o dell'array specificati
IsSupersetOf	Determina se il set corrente è un superset della raccolta o dell'array specificati

Metodo	Descrizione
Overlaps	Determina se il set corrente si sovrappone alla raccolta o array specificati
SetEquals	Determina se il set corrente contiene tutti gli elementi della raccolta o array specificati

ExceptWith

Produce l'operazione di differenza tra la raccolta corrente es una raccolta passata (array). Rimuove dall'insieme corrente (array) tutti gli elementi presenti nella raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void ExceptWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta da escludere dall'attuale set.

&collection[]

[in] Un array da escludere dal set corrente.

Note

Il risultato viene scritto nella raccolta corrente (array).

IntersectWith

Produce l'operazione dell'intersezione della raccolta corrente e di una raccolta passata (array). Modifica la raccolta corrente per contenere solo elementi presenti nella raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void IntersectWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta con cui verrà intersecato il set corrente.

&collection[]

[in] Un array con cui verrà intersecato il set corrente.

Note

Il risultato viene scritto nella raccolta corrente (array).

SymmetricExceptWith

Produce l'operazione di differenza simmetrica tra la raccolta corrente ed una raccolta passata (array). Modifica la raccolta corrente per contenere solo elementi presenti nell'oggetto sorgente o nella raccolta specificata (array), ma non in entrambi essi.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void SymmetricExceptWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta con cui produrre la differenza simmetrica .

&collection[]

[in] Un array con cui produrre la differenza simmetrica.

Note

Il risultato viene scritto nella raccolta corrente (array).

UnionWith

Produce l'unione della raccolta corrente ed una raccolta passata (array). Aggiunge agli elementi mancanti della raccolta corrente (array) dalla raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void UnionWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] A collection with which the current set will be united.

&collection[]

[in] Un array con cui il set corrente verrà unito.

Note

Il risultato viene scritto nella raccolta corrente (array).

IsProperSubsetOf

Determina se il set corrente è un sottoinsieme appropriato della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsProperSubsetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set corrente è un subset appropriato, altrimenti false.

IsProperSupersetOf

Determina se il set corrente è un superset appropriato della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsProperSupersetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se l'insieme corrente è un superset appropriato, altrimenti false.

IsSubsetOf

Determina se il set corrente è un subset della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsSubsetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set corrente è un subset, altrimenti false.

IsSupersetOf

Determina se il set corrente è un superset della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsSupersetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set corrente è un superset, altrimenti false .

Overlaps

Determina se il set corrente si sovrappone alla raccolta o all'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // la raccolta da confrontare
   );
```

Una versione per lavorare con un array.

```
bool Overlaps(
   T&  array[]                     // l'array da confrontare
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la sovrapposizione.

&collection[]

[in] Un array per determinare la sovrapposizione.

Valore di ritorno

Restituisce true se il set corrente ed una raccolta o un array si sovrappongono, altrimenti false.

SetEquals

Determina se il set corrente contiene tutti gli elementi della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // la raccolta da confrontare
   );
```

Una versione per lavorare con un array.

```
bool SetEquals(
   T&  array[]                     // l'array da confrontare
   );
```

Parametri

**collection*

[in] Una raccolta per confrontare elementi.

&collection[]

[in] Una raccolta per confrontare elementi.

Valore di ritorno

Restituisce true se il set corrente contiene tutti gli elementi della raccolta o dell'array specificati, altrimenti false.

CDefaultComparer<T>

CDefaultComparer<T> è una classe helper che implementa l'interfaccia IComparer<T> generica basata sui metodi globali Compare.

Descrizione

La classe CDefaultComparer<T> viene utilizzata per impostazione predefinita nelle raccolte di dati generiche, a meno che l'utente implicitamente utilizzi un'altra classe che attua l'interfaccia IComparer<T>.

Dichiarazione

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

Header

```
#include <Generic\Internal\DefaultComparer.mqh>
```

Inheritance Hierarchy

IComparer

CDefaultComparer

Class Methods

Metodo	Descrizione
Compare	Confronta due valori di tipo T

Compare

Confronta due valori di tipo T.

```
int Compare(
   T  x,      // il primo valore
   T  y       // il secondo valore
   );
```

Parametri

x

[in] Il primo valore da confrontare.

y

[in] Il secondo valore da confrontare.

Valore di ritorno

Restituisce un numero che esprime il rapporto tra i due valori confrontati:

- se il risultato è inferiore a zero, x è inferiore a y (x<y)
- se il risultato è uguale a zero, x è uguale a y (x=y)
- se il risultato è maggiore di zero, x è maggiore di y (x>y)

Note

I valori x e y vengono confrontati in base a uno degli overloads del metodo Global Compare a seconda del tipo T.

CDefaultEqualityComparer<T>

CDefaultEqualityComparer <T> è una classe helper che implementa l'interfaccia IEqualityComparer<T> generica basata su Equals<T> ed i metodi globali GetHashCode.

Descrizione

La classe CDefaultEqualityComparer<T> viene utilizzata per impostazione predefinita nelle raccolte di dati generiche, a meno che l'utente implicitamente utilizzi un'altra classe che implementa l'interfaccia IEqualityComparer<T>.

Dichiarazione

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

Header

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

Inheritance Hierarchy

IEqualityComparer

CDefaultEqualityComparer

Class Methods

Metodo	Descrizione
Equals	Confronta due valori di tipo T
HashCode	Calcola il valore del codice hash basato sull'oggetto di tipo T

Equals

Confronta due valori di tipo T.

```
bool Equals(
   T  x,      // il primo valore
   T  y       // il secondo valore
   );
```

Parametri

x

[in] Il primo valore da confrontare.

y

[in] Il secondo valore da confrontare.

Valore di ritorno

Restituisce true se i valori sono uguali, altrimenti false.

HashCode

Calcola il valore del codice hash basato sull'oggetto tipo T.

```
int HashCode(
   T  value     // un oggetto per il calcolo
   );
```

Parametri

value

[in] L'oggetto per cui si desidera ottenere il codice hash.

Valore di ritorno

Restituisce il codice hash.

CRedBlackTreeNode<T>

CRedBlackTreeNode<T> è una di helper utilizzata per implementare la classe CRedBlackTree<T>.

Descrizione

La classe CRedBlackTreeNode<T> è un nodo di CRedBlackTree<T> . I metodi di navigazione dell'albero vengono implementati nella classe.

Dichiarazione

```
template<typename T>
class CRedBlackTreeNode
```

Header

```
#include <Generic\RedBlackTree.mqh>
```

Class Methods

Metodo	Descrizione
Value	Restituisce ed imposta un valore di nodo
Parent	Restituisce ed imposta un puntatore al nodo principale
Left	Restituisce ed imposta un puntatore nel nodo sinistro
Right	Restituisce ed imposta un puntatore nel nodo destro
Color	Restituisce ed imposta un colore del nodo
IsLeaf	Determina se il nodo specificato è una foglia
CreateEmptyNode	Crea un nuovo nodo nero senza genitore e figli e restituisce un puntatore ad esso

Value (metodo Get)

Restituisce il valore del nodo.

```
T Value();
```

Valore di ritorno

Restituisce il valore del nodo.

Value (metodo Set)

Imposta il valore del nodo

```
void Value(
   T  value     // valore del nodo
   );
```

Parametri

value

[in] Valore del nodo.

Parent (metodo Get)

Restituisce un puntatore al nodo genitore.

```
CRedBlackTreeNode<T>* Parent();
```

Valore di ritorno

Restituisce un puntatore al nodo genitore.

Parent (metodo Set)

Imposta un puntatore al nodo genitore.

```
void Parent(
   CRedBlackTreeNode<T>*  node     // il puntatore al nodo genitore
   );
```

Parametri

node

[in] Un puntatore al nodo genitore.

Left (metodo Get)

Restituisce un puntatore al nodo sinistro.

```
CRedBlackTreeNode<T>* Left();
```

Valore di ritorno

Restituisce un puntatore al nodo sinistro.

Left (metodo Set)

Imposta un puntatore al nodo sinistro.

```
void Left(
   CRedBlackTreeNode<T>*  node     // un puntatore al nodo sinistro
   );
```

Parametri

node

[in] Un puntatore al nodo sinistro.

Right (metodo Get)

Restituisce un puntatore al nodo destro.

```
CRedBlackTreeNode<T>* Right();
```

Valore di ritorno

Restituisce un puntatore al nodo destro.

Right (metodo Set)

Imposta un puntatore al nodo destro.

```
void Right(
   CRedBlackTreeNode<T>*  node     // un puntatore al nodo destro
   );
```

Parametri

node

[in] Un puntatore al nodo destro.

Color (metodo Get)

Restituisce un colore del nodo.

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

Valore di ritorno

Restituisce un colore del nodo.

Color (metoto Set)

Imposta il colore del nodo.

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // colore del nodo
   );
```

Parametri

clr

[in] Colore del nodo.

Note

Il colore del nodo viene impostato utilizzando un valore da ENUM_RED_BLACK_TREE_NODE_TYPE. Può essere di due tipi:

- RED_BLACK_TREE_NODE_RED — il colore rosso del nodo;
- RED_BLACK_TREE_NODE_BLACK — il colore nero del nodo.

IsLeaf

Determina se il nodo specificato è una foglia.

```
bool IsLeaf();
```

Valore di ritorno

Restituisce true se il nodo è una foglia, altrimenti false.

CreateEmptyNode

Crea un nuovo nodo nero senza genitore e figli e restituisce un puntatore ad esso.

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

Valore di ritorno

Restituisce un puntatore al nuovo nodo.

CLinkedListNode<T>

CLinkedListNode<T> è una classe di helper utilizzata per implementare la classe CLinkedListNode<T>.

Descrizione

La classe CLinkedListNode<T> è un nodo dell'elenco CLinkedListNode duplicato<T> . I metodi di navigazione lista sono implementati nella classe.

Dichiarazione

```
template<typename T>
class CLinkedListNode
```

Header

```
#include <Generic\LinkedList.mqh>
```

Class Methods

Metodo	Descrizione
List	Restituisce ed imposta un puntatore a CLinkedList<T>
Next	Restituisce ed imposta un puntatore al nodo successivo
Previous	Restituisce e imposta un puntatore al nodo precedente
Value	Restituisce ed imposta il valore del nodo

List (il metodo Get)

Restituisce un puntatore a CLinkedList<T> .

```
CLinkedList<T>* List();
```

Valore di ritorno

Restituisce un puntatore alla lista collegata CLinkedList<T>.

List (metodo Set)

Sets a pointer to the CLinkedList<T>.

```
void List(
   CLinkedList<T>*  value     // un puntatore alla lista
   );
```

Parametri

value

[in] Un puntatore alla lista collegataCLinkedList<T>.

Next (metodo Get)

Restituisce un puntatore al nodo successivo.

```
CLinkedListNode<T>* Next();
```

Valore di ritorno

Restituisce un puntatore al nodo successivo.

Next (metodo Set)

Imposta un puntatore al nodo successivo

```
void Next(
   CLinkedListNode<T>*  value     // un puntatore al nodo successivo
   );
```

Parametri

value

[in] Un puntatore al nodo successivo.

Previous (metodo Get)

Restituisce un puntatore al nodo precedente.

```
CLinkedListNode<T>* Previous();
```

Valore di ritorno

Restituisce un puntatore al nodo precedente.

Precedente (metodo Set)

Imposta un puntatore al nodo precedente.

```
void Previous(
   CLinkedListNode<T>*  value     // un puntatore al nodo precedente
   );
```

Parametri

value

[in] Un puntatore al nodo precedente.

Value (metodo Get)

Restituisce il valore del nodo.

```
T Value();
```

Valore di ritorno

Restituisce il valore del nodo.

Value (metodo Set)

Imposta il valore del nodo

```
void Value(
   T  value     // Valore del nodo
   );
```

Parametri

value

[in] Valore del nodo.

CKeyValuePair<TKey, TValue>

La classe CKeyValuePair<TKey,TValue> implementa una coppia chiave/valore.

Descrizione

La classe CKeyValuePair<TKey,TValue> implementa metodi per lavorare con la chiave ed il valore della coppia chiave/valore.

Dichiarazione

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

Header

```
#include <Generic\HashMap.mqh>
```

Inheritance Hierarchy

IEqualityComparable

IComparable

CKeyValuePair

Class Methods

Metodo	Descrizione
Key	Ottiene ed imposta la chiave nella coppia chiave/valore
Value	Ottiene ed imposta il valore nella coppia chiave/valore
Clone	Crea una nuova coppia chiave/valore la cui chiave e valore sono uguali a quelli attuali
Compare	Confronta la coppia chiave/valore corrente con quella specificata
Equals	Controlla se la coppia corrente chiave/valore e quella specificata sono uguali
HashCode	Calcola il valore hash basato sulla coppia chiave/valore

Chiave (metodo Get)

Ottiene la chiave nella coppia chiave/valore.

```
TKey Key();
```

Valore di ritorno

Restituisce la chiave.

Key (metodo Set)

Imposta la chiave nella coppia chiave/valore.

```
void Key(
   TKey  key     // chiave
   );
```

Parametri

key

[in] Chiave.

Value (metodo Get)

Ottiene il valore nella coppia chiave/valore.

```
TValue Value();
```

Valore di ritorno

Restituisce il valore.

Value (metodo Set)

Imposta il valore nella coppia chiave/valore.

```
void Value(
   TValue  value     // valore
   );
```

Parametri

value

[in] Valore

Clone

Crea una nuova coppia chiave/valore la cui chiave e valore sono uguali a quelli attuali.

```
TValue>* Clone();
```

Valore di ritorno

Restituisce una nuova coppia chiave/valore

Compare

Confronta la coppia corrente chiave/valore con quella specificata.

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // la coppia da confrontare
   );
```

Parametri

pair

[in] La coppia da confrontare.

Valore di ritorno

Restituisce un numero che esprime il rapporto tra le coppie di valori correnti e passati:

- se il risultato è inferiore a zero, la coppia corrente chiave/valore è inferiore a quella passata
- se il risultato è zero, la coppia chiave/valore è uguale a quella passata
- se il risultato è maggiore di zero, la coppia corrente chiave/valore è maggiore di quella passata

Note

Le coppie chiave/valore vengono confrontate in base alle loro chiavi.

Equals

Controlla se la coppia corrente chiave/valore e quella specificata sono uguali.

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // la coppia da confrontare
   );
```

Parametri

pair

[in] La coppia da confrontare

Valore di ritorno

Restituisce true se le coppie chiave/valore sono uguali, altrimenti false.

Note

Le coppie chiave/valore vengono confrontate in base alle loro chiavi.

HashCode

Calcola il valore hash basato sulla coppia chiave/valore.

```
int HashCode();
```

Valore di ritorno

Restituisce il codice hash.

Note

Il codice hash della coppia chiave/valore è uguale al codice hash della chiave.

CArrayList<T>

CArrayList<T> è una classe generica che implementa l'interfaccia IList<T>.

Descrizione

La classe CArrayList<T> è un'implementazione dell'elenco dati dinamico di tipo T. Questa classe fornisce i metodi di base per lavorare con l'elenco, come accedere ad un elemento per indice, cercare ed eliminare elementi, ordinare e altro.

Dichiarazione

```
template<typename T>
class CArrayList : public IList<T>
```

Header

```
#include <Generic\ArrayList.mqh>
```

Inheritance Hierarchy

ICollection
  IList
    CArrayList

Class Methods

Metodo	Descrizione
Capacity	Ottiene ed Imposta la capacità corrente di un elenco
Count	Restituisce il numero di elementi nell'elenco
Contains	Determina se un elenco contiene un elemento con il valore specificato
TrimExcess	Imposta la capacità di un elenco al numero effettivo di elementi
TryGetValue	Ottiene un elemento dell'elenco all'indice specificato
TrySetValue	Imposta il valore dell'elemento dell' elenco nell'indice specificato
Add	Aggiunge un elemento all'elenco
AddRange	Aggiunge una raccolta o una serie di elementi all'elenco
Insert	Inserisce un elemento nell'elenco nell'indice specificato
InsertRange	Inserisce una raccolta o una serie di elementi nell'elenco dell'indice specificato
CopyTo	Copia tutti gli elementi di un elenco nell'array specificato partendo dall'indice specificato
BinarySearch	Cerca il valore specificato in un elenco ordinato in ordine crescente
IndexOf	Cerca la prima occorrenza di un valore in un elenco

Metodo	Descrizione
LastIndexOf	Cerca l'ultima occorrenza di un valore in un elenco
Clear	Rimuove tutti gli elementi da una raccolta
Remove	Rimuove la prima occorrenza dell'elemento specificato dall'elenco
RemoveAt	Rimuove un elemento nell'indice specificato dell'elenco
RemoveRange	Rimuove una serie di elementi dall'elenco
Reverse	Inverte l'ordine di elementi nell'elenco
Sort	Ordina gli elementi nell'elenco

Capacità (metodo Get)

Restituisce la capacità corrente della lista.

```
int Capacity();
```

Valore di ritorno

Restituisce la capacità corrente della lista.

Capacità (metodo Set)

Imposta la capacità corrente di un elenco.

```
void Capacity(
   const int  capacity     // valore della capacità
   );
```

Parametri

capacity

[in] Un nuovo valore di capacità.

Count

Restituisce il numero di elementi nell'elenco.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Contains

Determina se l'elenco contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nell'elenco, altrimenti false.

TrimExcess

Imposta la capacità di un elenco al numero effettivo di elementi e quindi libera la memoria inutilizzata.

```
void TrimExcess();
```

TryGetValue

Ottiene un elemento dell'elenco all'indice specificato.

```
bool TryGetValue(
   const int  index,      // indice
   T&         value       // una variabile da scrivere
   );
```

Parametri

index

[in] L'indice dell'elemento dell' elenco cui si desidera ottenere il valore.

&value

[out] La variabile per scrivere il valore dell'elemento.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TrySetValue

Imposta il valore dell'elemento dell' elenco nell'indice specificato.

```
bool TrySetValue(
   const int  index,      // indice
   T          value       // valore dell'elemento
   );
```

Parametri

index

[in] L'indice dell'elemento dell' elenco cui si desidera impostare il valore.

value

[in] Imposta il valore dell'elemento dell'elenco.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Add

Aggiunge un elemento all'elenco.

```
bool Add(
   T  value     // il valore dell'elemento
   );
```

Parametri

value

[in] Il valore dell'elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

AddRange

Aggiunge una raccolta o una serie di elementi all'elenco.

La versione che aggiunge un array.

```
bool AddRange(
   const T&  array[]               // un array da aggiungere
   );
```

La versione che aggiunge una raccolta.

```
bool AddRange(
   ICollection<T>*  collection     // una raccolta da aggiungere
   );
```

Parametri

&array[]

[in] Un array da aggiungere.

**collection*

[in] Una raccolta da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Insert

Inserisce un elemento nell'elenco nell'indice specificato.

```
bool Insert(
   const int  index,      // indice dell'insert
   T          item        // il valore da inserire
   );
```

Parametri

index

[in] L'indice in cui inserire.

item

[in] Il valore da inserire nell'indice specificato.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

InsertRange

Inserisce una raccolta o una serie di elementi nell'elenco dell'indice specificato.

La versione che inserisce un array.

```
bool InsertRange(
   const int  index,               // indice in cui inserire
   const T&   array[]              // l'array da inserire
   );
```

La versione che inserisce una raccolta.

```
bool InsertRange(
   const int        index,         // indice in cui inserire
   ICollection<T>*  collection     // la raccolta da inserire
   );
```

Parametri

index

[in] L'indice in cui inserire.

&array[]

[in] Un array da inserire nell'indice specificato.

**collection*

[in] Una raccolta da inserire nell'indice specificato.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CopyTo

Copia tutti gli elementi di un elenco nell'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // l'indice d'inizio per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritti gli elementi della lista.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

BinarySearch

Cerca il valore specificato in un elenco ordinato in ordine crescente.

La versione che esegue ricerche nell'intervallo di valori specificato utilizzando la classe che implementa l'interfaccia IComparable<T> per l'analisi degli elementi.

```
int BinarySearch(
   const int       index,        // l'indice iniziale
   const int       count,        // il raggio di ricerca
   T               item,         // il valore di ricerca
   IComparer<T>*   comparer      // interfaccia da confrontare
   );
```

La versione che esegue ricerche utilizzando la classe che implementa l'interfaccia IComparable<T> per l'analisi degli elementi.

```
int BinarySearch(
   T               item,         // il valore ricercato
   IComparer<T>*   comparer      // interfaccia da confrontare
   );
```

La versione che esegue ricerche utilizzando il metodo globale ::Compare per confrontare gli elementi.

```
int BinarySearch(
   T  item                       // il valore di ricerca
   );
```

Parametri

index

[in] L'indice d'inizio dal quale inizia la ricerca.

count

[in] La lunghezza del range di ricerca.

item

[in] Il valore cercato.

**comparer*

[in] Un interfaccia per confrontare elementi.

Valore di ritorno

Restituisce l'indice dell'elemento trovato. Se il valore di ricerca non viene trovato, restituisce l'indice dell'elemento più piccolo, che è il valore più vicino.

IndexOf

Cerca la prima occorrenza di un valore in un elenco.

Versione che cerca nell'intero elenco.

```
int IndexOf(
   T  item                    // il valore di ricerca
   );
```

Versione che cerca dalla posizione specificata e alla fine dell'elenco.

```
int IndexOf(
   T           item,          // il valore cercato
   const int   start_index    // l'indice iniziale
   );
```

Versione che ricerca dalla posizione specificata nell'intervallo specificato.

```
int IndexOf(
   T           item,          // il valore cercato
   const int   start_index,   // l'indice iniziale
   const int   count          // il raggio di ricerca
   );
```

Parametri

item

[in] Il valore cercato.

start_index

[in] L'indice d'inizio dal quale inizia la ricerca.

count

[in] La lunghezza del range di ricerca.

Valore di ritorno

Restituisce l'indice del primo elemento trovato. Se il valore non viene trovato, restituisce -1.

LastIndexOf

Cerca l'ultima occorrenza di un valore nell'elenco.

Versione che cerca nell'intero elenco.

```
int LastIndexOf(
   T  item                    // il valore di ricerca
   );
```

Versione che cerca dalla posizione specificata e alla fine dell'elenco.

```
int LastIndexOf(
   T          item,           // il valore cercato
   const int  start_index     // l'indice iniziale
   );
```

Versione che ricerca dalla posizione specificata nell'intervallo specificato.

```
int LastIndexOf(
   T          item,           // il valore cercato
   const int  start_index,    // l'indice iniziale
   const int  count           // il raggio di ricerca
   );
```

Parametri

item

[in] Il valore cercato.

start_index

[in] L'indice d'inizio dal quale inizia la ricerca.

count

[in] La lunghezza del range di ricerca.

Valore di ritorno

Restituisce l'indice dell'ultimo elemento trovato. Se il valore non viene trovato, restituisce -1.

Clear

Rimuove tutti gli elementi di una raccolta.

```
void Clear();
```

Remove

Rimuove la prima occorrenza dell'elemento specificato dall'elenco.

```
bool Remove(
   T  item     // il valore dell'elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

RemoveAt

Rimuove un elemento nell'indice specificato dell'elenco.

```
bool RemoveAt(
   const int  index     // indice
   );
```

Parametri

index

[in] L'indice dell'elemento da rimuovere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

RemoveRange

Rimuove una serie di elementi dall'elenco.

```
bool RemoveRange(
   const int  start_index,     // l'indice iniziale
   const int  count            // il numero di elementi
   );
```

Parametri

start_index

[in] L'indice di partenza da cui inizia l'eliminazione.

count

[in] Il numero di elementi da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Reverse

Inverte l'ordine di elementi nell'elenco.

La versione per lavorare con l'intera lista.

```
bool Reverse();
```

La versione per il funzionamento con l'intervallo di elementi di elenco specificato.

```
bool Reverse(
   const int  start_index,     // l'indice iniziale
   const int  count            // il numero di elementi
   );
```

Parametri

start_index

[in] L'indice d'inizio.

count

[in] Il numero di elementi dell' elenco che partecipano all'operazione.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Sort

Ordina gli elementi nell'elenco.

La versione che ordina tutti gli elementi dell'elenco.

```
bool Sort();
```

La versione che ordina tutti gli elementi nell'elenco utilizzando la classe che implementa l'interfaccia IComparable<T> per l'analisi degli elementi.

```
bool Sort(
   IComparer<T>*  comparer         // interfaccia per il confronto
   );
```

La versione che ordina l'intervallo di elementi specificato nell'elenco utilizzando la classe che implementa l'interfaccia IComparable<T> per l'analisi degli elementi.

```
bool Sort(
   const int  start_index,     // l'indice d'inizio
   const int  count            // il numero di elementi
   IComparer<T>*  comparer     // interfaccia da confrontare
   );
```

Parametri

**comparer*

[in] Un interfaccia per confrontare elementi.

start_index

[in] L'indice di partenza da cui inizia l'ordinamento.

count

[in] La lunghezza dell'intervallo di selezione.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CHashMap<TKey, TValue>

CHashMap<TKey, TValue> è una classe generica che implementa l'interfaccia IMap<TKey, TValue>.

Descrizione

La clase CHashMap<TKey, TValue> è un'implementazione della tabella hash dinamica, i cui dati vengono memorizzati sottoforma di coppie di chiave/valore non ordinate tenendo conto del requisito di unicità chiave. Questa classe fornisce metodi di base per lavorare con una tabella hash, ad esempio per accedere ad un valore tramite chiave, per cercare ed eliminare una coppia di chiave/valore e altro.

Dichiarazione

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey, TValue>
```

Header

```
#include <Generic\HashMap.mqh>
```

Inheritance Hierarchy

ICollection

IMap

CHashMap

Class Methods

Metodo	Descrizione
Add	Aggiunge una coppia di chiave/valore alla tabella hash
Count	Restituisce il numero di elementi nella tabella hash
Comparer	Restituisce un puntatore all'interfaccia IEqualityComparer<T>, utilizzata per organizzare una tabella hash
Contains	Determina se la tabella hash contiene la coppia di chiave/valore specificata
ContainsKey	Determina se la tabella hash contiene la coppia di chiave/valore con la chiave specificata
ContainsValue	CHashMap<TKey, TValue> è una classe generica che implementa l'interfaccia IMap<TKey, TValue>
CopyTo	Copia tutte le coppie chiave/valore dalla tabella hash agli array specificati, a partire dall'indice specificato
Clear	Rimuove tutti gli elementi dalla tabella hash
Remove	Rimuove la prima occorrenza della coppia di chiave/valore dalla tabella hash

Metodo	Descrizione
TryGetValue	Ottiene un elemento con la chiave specificata dalla tabella hash
TrySetValue	Modifica il valore di una coppia di chiave/valore dalla tabella hash alla chiave specificata

Add

Aggiunge una coppia di chiave/valore alla tabella hash.

Una versione che aggiunge una coppia di chiave/valore generata.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // la coppia chiave/valore
   );
```

Una versione che aggiunge una nuova coppia chiave/valore con la chiave e il valore specificati.

```
bool Add(
   TKey    key,                         // chiave
   TValue  value                        // valore
   );
```

Parametri

pair

[in] La coppia chiave/valore.

key

[in] Chiave.

value

[in] Valore.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi nella tabella hash.

```
int Count();
```

Comparer

Restituisce un puntatore all'interfaccia IEqualityComparer<T>, utilizzata per organizzare una tabella hash.

```
IEqualityComparer<TKey>* Comparer() const;
```

Valore di ritorno

Restituisce un puntatore all'interfaccia IEqualityComparer<T>.

Contains

Determina se la tabella hash contienela coppia specificata chiave/valore.

La versione per lavorare con una coppia chiave/valore generata.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // la coppia chiave/valore
   );
```

La versione per lavorare con una coppia chiave/valore sotto forma di chiave e valore impostati separatamente.

```
bool Contains(
   TKey   key,                          // chiave
   TValue value                         // valore
   );
```

Parametri

**item*

[in] La coppia chiave/valore.

key

[in] Chiave.

value

[in] Valore.

Valore di ritorno

Restituisce true, se la tabella hash contiene la coppia chiave/valore con il valore e chiave specificati, o altrimenti false.

ContainsKey

Determina se la tabella hash contienela coppia chiave/valore con la chiave specificata.

```
bool ContainsKey(
   TKey  key     // chiave
   );
```

Parametri

key

[in] Chiave.

Valore di ritorno

Restituisce true se la tabella hash contiene una coppia chiave/valore con la chiave specificata, o altrimenti false.

ContainsValue

Determina se la tabella hash contienela coppia di chiave/valore con il valore specificato.

```
bool ContainsValue(
   TValue  value     // valore
   );
```

Parametri

value

[in] Valore.

Valore di ritorno

Restituisce true se la tabella hash contiene una coppia di chiave/valore con il valore specificato, altrimenti false .

CopyTo

Copia tutte le coppie di chiave/valore dalla tabella hash agli array specificati, a partire dall'indice specificato.

La versione che copia una tabella hash alla matrice di coppie chiave/valore.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // un array per scrivere coppie chiav
   const int                    dst_start=0      // l'indice iniziale per scrivere
   );
```

La versione che copia una tabella hash per separare gli array per chiavi e valori.

```
int CopyTo(
   TKey&       dst_keys[],                       // un array per scrivere chiavi
   TValue&     dst_values[],                     // un array per scrivere valori
   const int   dst_start=0                       // indice iniziale per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritte tutte le coppie della tabella hash.

&dst_keys[]

[out] Un array nel quale verranno scritte tutte le chiavi della tabella hash.

&dst_values[]

[out] Un array nel quale verranno scritti tutti i valori della tabella hash.

dst_start=0

[in] L'indice di array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di coppie di chiave/valore copiate.

Clear

Rimuove tutti gli elementi dalla tabella hash.

```
void Clear();
```

Remove

Rimuove la prima occorrenza della coppia di chiave/valore dalla tabella hash.

La versione che rimuove una coppia chiave-valore in base alla coppia di chiave-valore generata.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // la coppia chiave/valore"
   );
```

La versione che rimuove una coppia chiave-valore in base alla chiave.

```
bool Remove(
   TKey  key                           // chiave
   );
```

Parametri

item

[in] La coppia chiave/valore.

key

[in] Chiave.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TryGetValue

Ottiene un elemento con la chiave specificata dal file tabella hash.

```
bool TryGetValue(
   TKey      key,       // chiave
   TValue&   value      // variabile per scrivere il valore
   );
```

Parametri

key

[in] Chiave.

&value

[out] La variabile a cui verrà scritto il valore specificato della coppia chiave/valore.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TrySetValue

Modifica il valore di una coppia di chiave/valore dalla tabella hash alla chiave specificata.

```
bool TrySetValue(
   TKey    key,       // chiave
   TValue  value      // nuovo valore
   );
```

Parametri

key

[in] Chiave.

value

[in] Il nuovo valore da assegnare alla coppia di chiave-valore specificata.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CHashSet<T>

CHashSet<T> è una classe generica che implementa l'interfaccia ISet<T>.

Descrizione

La classe CHashSet<T> è un'implementazione del set di dati dinamico non ordinato del tipo T, con l'unicità richiesta di ogni valore. Questa classe fornisce metodi di base per lavorare con set e operazioni correlate, quali: l'unione e l'intersezione di sets, la definizione di sub-sets stretti e non, ed altro.

Dichiarazione

```
template<typename T>
class CHashSet : public ISet<T>
```

Header

```
#include <Generic\HashSet.mqh>
```

Inheritance Hierarchy

ICollection
  ISet
    CHashSet

Class Methods

Metodo	Descrizione
Add	Aggiunge un elemento ad un set
Count	Restituisce il numero di elementi di un set
Comparer	Determina se un set contiene un elemento con il valore specificato
Contains	Restituisce un puntatore all'interfaccia IEqualityComparer<T>, utilizzata per organizzare un set
TrimExcess	Imposta la capacità di un set per il numero effettivo di elementi e quindi libera la memoria inutilizzata
CopyTo	Copia tutti gli elementi di un set all'array specificato partendo dall'indice specificato
Clear	Rimuove tutti gli elementi da un set
Remove	Rimuove l'elemento specificato da un set
ExceptWith	Produce l'operazione di differenza tra la raccolta corrente e una raccolta passata (array)
IntersectWith	Produce il funzionamento dell'intersezione della raccolta corrente ed una raccolta passata (array)

Metodo	Descrizione
SymmetricExceptWith	Produce l'operazione di differenza simmetrica tra la raccolta corrente ed una raccolta passata (array)
UnionWith	Produce l'unione della raccolta corrente ed una raccolta passata (array)
IsProperSubsetOf	Determina se il set corrente è un sottoinsieme appropriato della raccolta o dell'array specificati
IsProperSupersetOf	Determina se il set corrente è un superset appropriato della raccolta o array specificati
IsSubsetOf	Determina se il set corrente è un subset della raccolta o dell'array specificati
IsSupersetOf	Determina se il set corrente è un superset della raccolta o dell'array specificati
Overlaps	Determina se il set corrente si sovrappone alla raccolta o array specificati
SetEquals	Determina se il set corrente contiene tutti gli elementi della raccolta o array specificati

Add

Aggiunge un elemento ad un set.

```
bool Add(
   T  value     // il valore dell'elemento
   );
```

Parametri

value

[in] Il valore dell'elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi di un set.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Contains

Determina se un set contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nel set, altrimenti false.

Comparer

Restituisce un puntatore all'interfaccia IEqualityComparer<T>, utilizzata per organizzare un set.

```
IEqualityComparer<T>* Comparer() const;
```

Valore di ritorno

Restituisce un puntatore all'interfaccia IEqualityComparer<T>.

TrimExcess

Imposta la capacità di un set ad numero effettivo di elementi, e quindi libera la memoria inutilizzata.

```
void TrimExcess();
```

CopyTo

Copia tutti gli elementi di un set all'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // indice iniziale per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritti gli elementi del set.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

Clear

Rimuove tutti gli elementi da un set.

```
void Clear();
```

Remove

Rimuove l'elemento specificato da un set.

```
bool Remove(
   T  item     // il valore dell'elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

ExceptWith

Produce l'operazione di differenza tra la raccolta corrente es una raccolta passata (array). Rimuove dall'insieme corrente (array) tutti gli elementi presenti nella raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void ExceptWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta da escludere dall'attuale set.

&collection[]

[in] Un array da escludere dal set corrente.

Note

Il risultato viene scritto nella raccolta corrente (array).

IntersectWith

Produce l'operazione dell'intersezione della raccolta corrente e di una raccolta passata (array). Modifica la raccolta corrente per contenere solo elementi presenti nella raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void IntersectWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta con cui verrà intersecato il set corrente.

&collection[]

[in] Un array con cui verrà intersecato il set corrente.

Note

Il risultato viene scritto nella raccolta corrente (array).

SymmetricExceptWith

Produce l'operazione di differenza simmetrica tra la raccolta corrente ed una raccolta passata (array). Modifica la raccolta corrente per contenere solo elementi presenti nell'oggetto sorgente o nella raccolta specificata (array), ma non in entrambi essi.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void SymmetricExceptWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta con cui produrre la differenza simmetrica .

&collection[]

[in] Un array con cui produrre la differenza simmetrica.

Note

Il risultato viene scritto nella raccolta corrente (array).

UnionWith

Produce l'unione della raccolta corrente ed una raccolta passata (array). Aggiunge agli elementi mancanti della raccolta corrente (array) dalla raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void UnionWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] A collection with which the current set will be united.

&collection[]

[in] Un array con cui il set corrente verrà unito.

Note

Il risultato viene scritto nella raccolta corrente (array).

IsProperSubsetOf

Determina se il set corrente è un sottoinsieme appropriato della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsProperSubsetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set corrente è un subset appropriato, altrimenti false.

IsProperSupersetOf

Determina se il set corrente è un superset appropriato della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsProperSupersetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se l'insieme corrente è un superset appropriato, altrimenti false.

IsSubsetOf

Determina se il set corrente è un subset della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsSubsetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set corrente è un subset, altrimenti false.

IsSupersetOf

Determina se il set corrente è un superset della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsSupersetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set corrente è un superset, altrimenti false .

Overlaps

Determina se il set corrente si sovrappone alla raccolta o all'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // la raccolta da confrontare
   );
```

Una versione per lavorare con un array.

```
bool Overlaps(
   T&  array[]                     // l'array da confrontare
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la sovrapposizione.

&collection[]

[in] Un array per determinare la sovrapposizione.

Valore di ritorno

Restituisce true se il set corrente ed una raccolta o un array si sovrappongono, altrimenti false.

SetEquals

Determina se il set corrente contiene tutti gli elementi della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // la raccolta da confrontare
   );
```

Una versione per lavorare con un array.

```
bool SetEquals(
   T&  array[]                     // l'array da confrontare
   );
```

Parametri

**collection*

[in] Una raccolta per confrontare elementi.

&collection[]

[in] Un array per confrontare gli elementi.

Valore di ritorno

Restituisce true se il set corrente contiene tutti gli elementi della raccolta o dell'array specificati, altrimenti false.

CLinkedList<T>

CLinkedList <T> è una classe generica che implementa l'interfaccia ICollection<T>.

Descrizione

La classe CLinkedList<T> è un'implementazione dell'elenco dati dinamico doppiamente collegato di tipo T. Questa classe fornisce metodi di base per lavorare con liste doppiamente collegate, come ad esempio aggiungere, eliminare, cercare elementi e altro.

Dichiarazione

```
template<typename T>
class CLinkedList : public ICollection<T>
```

Header

```
#include <Generic\LinkedList.mqh>
```

Inheritance Hierarchy

ICollection

CLinkedList

Class Methods

Metodo	Descrizione
Add	Aggiunge un elemento ad un elenco collegato
AddAfter	Aggiunge un elemento dopo il nodo specificato nella lista collegata
AddBefore	Aggiunge un elemento prima del nodo specificato nella lista collegata
AddFirst	Aggiunge un elemento all'inizio della lista collegata
AddLast	Aggiunge un elemento alla fine della lista collegata
Count	Restituisce il numero di elementi nella lista collegata
Head	Restituisce un puntatore al primo nodo della lista collegata
First	Restituisce un puntatore al primo nodo della lista collegata
Last	Restituisce un puntatore all'ultimo nodo della lista collegata
Contains	Determina se la lista collegata contiene un elemento con il valore specificato
CopyTo	Copia tutti gli elementi della lista collegata all'array specificato partendo dall'indice specificato
Clear	Rimuove tutti gli elementi da una lista collegata
Remove	Rimuove la prima occorrenza dell'elemento specificato dalla lista collegata

Metodo	Descrizione
RemoveFirst	Rimuove il primo elemento della lista collegata
RemoveLast	Rimuove l'ultimo elemento della lista collegata
Find	Cerca la prima occorrenza del valore specificato nella lista collegata
FindLast	Cerca l'ultima occorrenza del valore specificato nella lista collegata

Add

Aggiunge un elemento ad una lista collegata.

```
bool Add(
   T  value     // il valore dell'elemento
   );
```

Parametri

value

[in] Il valore dell'elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

AddAfter

Aggiunge un elemento dopo il nodo specificato nella lista collegata.

La versione che aggiunge un elemento per valore.

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*   node,         // il dopo il quale l'elemento dev'essere aggiunt
   T                     value         // l'elemento da aggiungere
   );
```

Valore di ritorno

Restituisce un puntatore al nodo aggiunto.

La versione che aggiunge un elemento come un nodo formato per valore.

```
bool AddAfter(
   CLinkedListNode<T>*   node,         // il dopo il quale l'elemento dev'essere aggiunt
   CLinkedListNode<T>*   new_node      // il nodo da aggiungere
   );
```

Parametri

**node*

[in] Il nodo dalla lista collegata, dopo il quale verrà aggiunto un nuovo elemento.

value

[in] Un elemento da aggiungere.

**new_node*

[in] Un nodo da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

AddBefore

Aggiunge un elemento prima del nodo specificato nell'elenco collegato.

La versione che aggiunge un elemento per valore.

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*  node,         // il nodo prima del quale l'elemento dev'essere
   T                    value         // l'elemento da aggiungere
   );
```

Valore di ritorno

Restituisce un puntatore al nodo aggiunto.

La versione che aggiunge un elemento come un nodo formato per valore.

```
bool AddBefore(
   CLinkedListNode<T>*  node,         // il nodo prima del quale l'elemento dev'essere
   CLinkedListNode<T>*  new_node      // il nodo da aggiungere
   );
```

Parametri

node

[in] Il nodo della lista collegata, prima che verrà aggiunto un nuovo elemento.

value

[in] Un elemento da aggiungere.

**new_node*

[in] Un nodo da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

AddFirst

Aggiunge un elemento all'inizio della lista collegata.

La versione che aggiunge un elemento per valore.

```
CLinkedListNode<T>* AddFirst(
   T    value                    // l'elemento da aggiungere
   );
```

Valore di ritorno

Restituisce un puntatore al nodo aggiunto.

La versione che aggiunge un elemento come un nodo formato per valore.

```
bool AddFirst(
   CLinkedListNode<T>*  node     // il nodo da aggiungere
   );
```

Parametri

value

[in] Un elemento da aggiungere.

**node*

[in] Un nodo da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

AddLast

Aggiunge un elemento alla fine della lista collegata

La versione che aggiunge un elemento per valore.

```
CLinkedListNode<T>* AddLast(
   T  value                        // un elemento da aggiungere
   );
```

Valore di ritorno

Restituisce un puntatore al nodo aggiunto.

La versione che aggiunge un elemento come un nodo formato per valore.

```
bool AddLast(
   CLinkedListNode<T>*  node      // il nodo da aggiungere
   );
```

Parametri

value

[in] Un elemento da aggiungere.

**node*

[in] Un nodo da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi nella lista collegata.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Head

Restituisce un puntatore al primo nodo della lista collegata.

```
CLinkedListNode<T>* Head();
```

Valore di ritorno

Restituisce un puntatore al primo nodo.

First

Restituisce un puntatore al primo nodo della lista collegata.

```
CLinkedListNode<T>* First();
```

Valore di ritorno

Restituisce un puntatore al primo nodo.

Last

Restituisce un puntatore all'ultimo nodo della lista collegata.

```
CLinkedListNode<T>* Last();
```

Valore di ritorno

Restituisce un puntatore all'ultimo nodo.

Contains

Determina se l'elenco collegato contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nella lista collegata, altrimenti false.

CopyTo

Copia tutti gli elementi dell'elenco collegato all'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // l'indice d'inizio per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array al quale verranno scritti gli elementi della lista collegata.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

Clear

Rimuove tutti gli elementi di una raccolta.

```
void Clear();
```

Remove

Rimuove la prima occorrenza dell'elemento specificato dalla lista collegata.

La versione che rimuove un elemento per valore.

```
bool Remove(
   T  item                        // il valore dell'elelento
   );
```

La versione che rimuove un elemento dal puntatore ad un nodo.

```
bool Remove(
   CLinkedListNode<T>*  node      // il nodo dell'elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

**node*

[in] Il nodo dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

RemoveFirst

Rimuove il primo elemento della lista collegata.

```
bool RemoveFirst();
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

RemoveLast

Rimuove l'ultimo elemento della lista collegata.

```
bool RemoveLast();
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Find

Cerca la prima occorrenza del valore specificato nella lista collegata.

```
CLinkedListNode<T>* Find(
   T  value     // il valore ricercato
   );
```

Parametri

value

[in] Il valore cercato.

Valore di ritorno

Restituisce un puntatore al primo nodo trovato che contiene il valore di ricerca sul successo, altrimenti NULL.

FindLast

Cerca l'ultima occorrenza del valore specificato nell'elenco collegato.

```
CLinkedListNode<T>* FindLast(
   T  value     // il valore ricercato
   );
```

Parametri

value

[in] Il valore cercato.

Valore di ritorno

Restituisce un puntatore all'ultimo nodo trovato contenente il valore di ricerca in caso di successo, altrimenti NULL.

CQueue<T>

CQueue<T> è una classe generica che implementa l'interfaccia ICollection<T>.

Descrizione

La classe CQueue<T> è una raccolta dinamica di dati di tipo T, organizzata come una coda che opera sul principio FIFO (il primo ad entrare è il primo ad uscire).

Dichiarazione

```
template<typename T>
class CQueue : public ICollection<T>
```

Header

```
#include <Generic\Queue.mqh>
```

Inheritance Hierarchy

ICollection

CQueue

Class Methods

Metodo	Descrizione
Add	Aggiunge un elemento ad una coda
Enqueue	Aggiunge un elemento ad una coda
Count	Restituisce il numero di elementi nella coda
Contains	Determina se la coda contiene un elemento con il valore specificato
TrimExcess	Imposta la capacità di una coda al numero effettivo di elementi e quindi libera la memoria non utilizzata
CopyTo	Copia tutti gli elementi di una coda nell'array specificato partendo dall'indice specificato
Clear	Rimuove tutti gli elementi da una coda
Remove	Rimuove la prima occorrenza dell'elemento specificato dalla coda
Dequeue	Restituisce l'elemento di partenza e lo rimuove dalla coda
Peek	Restituisce l'elemento di partenza senza rimuoverlo dalla coda

Add

Aggiunge un elemento a una coda.

```
bool Add(
   T  value     // il valore dell'elemento
   );
```

Parametri

value

[in] Il valore dell'elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Enqueue

Aggiunge un elemento a una coda.

```
bool Enqueue(
   T  value     // elemento da aggiungere
   );
```

Parametri

value

[in] Un elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi nella coda.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Contains

Determina se la coda contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore ricercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nella coda, altrimenti false .

TrimExcess

Imposta la capacità di una coda al numero effettivo di elementi e quindi libera la memoria non utilizzata.

```
void TrimExcess();
```

CopyTo

Copia tutti gli elementi di una coda nell'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // l'indice d'inizio per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritti gli elementi della coda.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

Clear

Rimuove tutti gli elementi da una coda.

```
void Clear();
```

Remove

Rimuove la prima occorrenza dell'elemento specificato dalla coda.

```
bool Remove(
   T  item     // il valore dell'elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Dequeue

Restituisce il primo elemento e lo rimuove dalla coda.

```
T Dequeue();
```

Valore di ritorno

Restituisce l'elemento di partenza.

Peek

Restituisce il primo elemento senza rimuoverlo dalla coda.

```
T Peek();
```

Valore di ritorno

Restituisce l'elemento di partenza.

CRedBlackTree<T>

CRedBlackTree<T> è una classe generica che implementa l'interfaccia ICollection<T>.

Descrizione

La classe CRedBlackTree<T> è un'implementazione di un albero rosso-nero dinamico i cui nodi memorizzano i dati di tipo T. La classe fornisce metodi di base per lavorare con alberi rosso-neri, come ad esempio aggiungere, eliminare, cercare il valore massimo e minimo ed altro ancora.

Dichiarazione

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

Header

```
#include <Generic\RedBlackTree.mqh>
```

Inheritance Hierarchy

ICollection

CRedBlackTree

Class Methods

Metodo	Descrizione
Add	Aggiunge un elemento ad un albero rosso-nero
Root	Restituisce un puntatore alla radice dell'albero rosso-nero
Count	Restituisce il numero di elementi nell'albero rosso-nero
Contains	Determina se l'albero rosso-nero contiene un elemento con il valore specificato
Comparer	Restituisce un puntatore all'interfaccia IComparer<T> utilizzata per organizzare un albero rosso-nero
TryGetMin	Ottiene l'elemento minimo di un albero rosso-nero
TryGetMax	Ottiene l'elemento massimo di un albero rosso-nero
CopyTo	Copia tutti gli elementi di un albero rosso-nero all'array specificato partendo dall'indice specificato
Clear	Rimuove tutti gli elementi da un albero rosso-nero
Remove	Rimuove il verificarsi dell'elemento specificato da un albero rosso-nero
RemoveMin	Rimuove un elemento con il valore minimo da un albero rosso-nero
RemoveMax	Rimuove un elemento con il valore massimo da un albero rosso-nero
Find	Cerca il verificarsi di un valore specificato in un albero rosso-nero

Metodo	Descrizione
FindMax	Cerca un elemento con il valore massimo in un albero rosso-nero
FindMin	Ricerca di un elemento con il valore minimo in un albero rosso-nero

Add

Aggiunge un elemento ad un albero rosso-nero.

```
bool Add(
   T  value     // elemento da aggiungere
   );
```

Parametri

value

[in] Un elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi nell'albero rosso-nero.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Root

Restituisce un puntatore alla radice dell'albero rosso-nero.

```
CRedBlackTreeNode<T>* Root();
```

Valore di ritorno

Restituisce un puntatore alla radice.

Contains

Determina se l'albero rosso-nero contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nell'albero rosso-nero, altrimenti false.

Comparer

Restituisce un puntatore all'interfaccia IComparer<T> utilizzata per organizzare un albero rosso-nero.

```
IComparer<T>* Comparer() const;
```

Valore di ritorno

Restituisce un puntatore all'interfaccia IComparer<T>.

TryGetMin

Ottiene l'elemento minimo di un albero rosso-nero.

```
bool TryGetMin(
   T&  min     // la variabile per scrivere il valore
   );
```

Parametri

&min

[out] La variabile in cui verrà scritto il valore minimo.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TryGetMax

Ottiene l'elemento massimo di un albero rosso-nero.

```
bool TryGetMax(
   T&  max     // la variabile per scrivere il valore
   );
```

Parametri

&max

[out] La variabile in cui verrà scritto il valore massimo.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CopyTo

Copia tutti gli elementi di un albero rosso-nero all'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // indice iniziale per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritti gli elementi dell'albero rosso-nero.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

Clear

Rimuove tutti gli elementi da un albero rosso-nero.

```
void Clear();
```

Remove

Rimuove il verificarsi dell'elemento specificato da un albero rosso-nero.

La versione che rimuove un elemento con il valore specificato.

```
bool Remove(
   T  value                        // il valore dell' elemento
   );
```

La versione che rimuove un elemento dal puntatore ad un nodo.

```
bool Remove(
   CRedBlackTreeNode<T>*  node     // il nodo dell' elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

**node*

[in] Il nodo dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

RemoveMin

Rimuove un elemento con il valore minimo da un albero rosso-nero.

```
bool RemoveMin();
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

RemoveMax

Rimuove un elemento con il valore massimo da un albero rosso-nero.

```
bool RemoveMax();
```

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Find

Cerca il verificarsi di un valore specificato in un albero rosso-nero.

```
CRedBlackTreeNode<T>* Find(
   T  value     // il valore ricercato
   );
```

Parametri

value

[in] Il valore cercato.

Valore di ritorno

Restituisce un puntatore al nodo trovato contenente il valore di ricerca in caso di successo, altrimenti NULL.

FindMin

Ricerca di un elemento con il valore minimo in un albero rosso-nero.

```
CRedBlackTreeNode<T>* FindMin();
```

Valore di ritorno

Restituisce un puntatore al nodo contenente il valore minimo in caso di successo, altrimenti NULL.

FindMax

Cerca un elemento con il valore massimo in un albero rosso-nero.

```
CRedBlackTreeNode<T>* FindMax();
```

Valore di ritorno

Restituisce un puntatore al nodo contenente il valore massimo in caso di successo, altrimenti NULL.

CSortedMap<TKey, TValue>

CSortedMap<TKey,TValue> è una classe generica che implementa l'interfaccia IMap<TKey,TValue>.

Descrizione

La classe CSortedMap<TKey,TValue> è un'implementazione di una tabella hash dinamica i cui dati vengono memorizzati come coppie chiave/valore ordinate per chiave e tenendo conto del requisito di unicità chiave. Questa classe fornisce metodi di base per lavorare con una tabella hash, ad esempio per accedere ad un valore tramite chiave, per cercare ed eliminare una coppia di chiave/valore e altro.

Dichiarazione

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

Header

```
#include <Generic\SortedMap.mqh>
```

Inheritance Hierarchy

ICollection
  IMap
    CSortedMap

Class Methods

Metodo	Descrizione
Add	Aggiunge una coppia di chiave/valore alla tabella hash
Count	Restituisce il numero di elementi nella tabella di hash ordinata
Contains	Determina se la tabella hash ordinata contiene la tabella chiave/valore specificata
ContainsKey	Determina se la tabella hash ordinata contiene la tabella chiave/valore con la chiave specificata
ContainsValue	Determina se la tabella hash ordinata contiene la tabella chiave/valore con il valore specificato
Comparer	Restituisce un puntatore all'interfaccia IComparer<T>, utilizzata per organizzare una tabella di hash ordinata
CopyTo	Copia tutte le coppie chiave/valore dalla tabella hash ordinata agli array specificati, a partire dall'indice specificato
Clear	Rimuove tutti gli elementi dalla tabella hash ordinata
Remove	Rimuove la prima occorrenza della coppia chiave/valore dalla tabella hash ordinata

Metodo	Descrizione
TryGetValue	Ottiene un elemento con la chiave specificata dalla tabella hash ordinata
TrySetValue	Modifica una coppia di chiave/valore con la chiave specificata dalla tabella hash ordinata

Add

Aggiunge una coppia di chiave/valore alla tabella hash.

Una versione che aggiunge una coppia di chiave/valore generata.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // la coppia chiave/valore
   );
```

Una versione che aggiunge una nuova coppia chiave/valore con la chiave e il valore specificati.

```
bool Add(
   TKey    key,                         // chiave
   TValue  value                        // valore
   );
```

Parametri

pair

[in] La coppia chiave/valore.

key

[in] Chiave.

value

[in] Valore.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi nella tabella hash ordinata.

```
int Count();
```

Comparer

Restituisce un puntatore all'interfaccia IComparer<T>, utilizzata per organizzare una tabella hash ordinata.

```
IComparer<TKey>* Comparer() const;
```

Valore di ritorno

Restituisce un puntatore all'interfaccia IComparer<T>.

Contains

Determina se la tabella hashordinata contiene la tabella chiave/valore specificata.

La versione per lavorare con una coppia chiave/valore generata.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // la coppia chiave/valore
   );
```

La versione per lavorare con una coppia chiave/valore sotto forma di chiave e valore impostati separatamente.

```
bool Contains(
   TKey    key,                         // chiave
   TValue  value                        // valore
   );
```

Parametri

item

[in] La coppia chiave/valore.

key

[in] Chiave.

value

[in] Valore.

Valore di ritorno

Restituisce true, se la tabella hash ordinata (hash table) contiene la coppia di chiave/valore con la chiave e il valore specificati, altrimenti false.

ContainsKey

Determina la tabella hash ordinata contiene la tabella chiave/valore con la chiave specificata.

```
bool ContainsKey(
   TKey  key     // chiave
   );
```

Parametri

key

[in] Chiave.

Valore di ritorno

Restituisce true, se la tabella hash ordinata contiene la coppia di chiave/valore con la chiave specificata, altrimenti false.

ContainsValue

Determina se la tabella hash ordinata contiene la chiave/valore con il valore specificato.

```
bool ContainsValue(
   TValue  value     // valore
   );
```

Parametri

value

[in] Valore.

Valore di ritorno

Restituisce true, se la tabella hash ordinata contiene la coppia chiave/valore con il valore specificato, altrimenti false.

CopyTo

Copia tutte le coppie chiave/valore dalla tabella hash ordinata in array specificati, a partire dall'indice specificato.

La versione che copia una tabella hash alla matrice di coppie chiave/valore.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // l'array per la scrittura di coppie
   const int                    dst_start=0      // l'indice iniziale per scrivere
   );
```

La versione che copia una tabella hash per separare gli array per chiavi e valori.

```
int CopyTo(
   TKey&       dst_keys[],                       // un array per scrivere le chiavi
   TValue&     dst_values[],                     // un array per scrivere i valori
   const int   dst_start=0                       // l'indice iniziale per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritte tutte le coppie della tabella hash.

&dst_keys[]

[out] Un array nel quale verranno scritte tutte le chiavi della tabella hash.

&dst_values[]

[out] Un array nel quale verranno scritti tutti i valori della tabella hash.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di coppie di chiave/valore copiate.

Clear

Rimuove tutti gli elementi dalla tabella hash ordinata.

```
void Clear();
```

Remove

Rimuove la prima occorrenza della coppia di chiave/valore dalla tabella hash ordinata.

La versione che rimuove una coppia chiave-valore in base alla coppia di chiave-valore generata.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // la coppia chiave/valore
   );
```

La versione che rimuove una coppia chiave-valore in base alla chiave.

```
bool Remove(
   TKey  key                           // chiave
   );
```

Parametri

**item*

[in] La coppia chiave/valore.

key

[in] Chiave.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TryGetValue

Ottiene un elemento con la chiave specificata dalla tabella hash ordinata.

```
bool TryGetValue(
   TKey      key,       // chiave
   TValue&   value      // uan variabile per scrivere il valore
   );
```

Parametri

key

[in] Chiave.

&value

[out] La variabile a cui verrà scritto il valore specificato della coppia chiave/valore.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TrySetValue

Modifica il valore della coppia chiave/valore della tabella hash ordinata, alla chiave specificata.

```
bool TrySetValue(
   TKey    key,       // chiave
   TValue  value      // nuovo valore
   );
```

Parametri

key

[in] Chiave.

value

[in] Il nuovo valore da assegnare alla coppia di chiave-valore specificata.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CSortedSet<T>

CSortedSet<T> è una classe generica che implementa l'interfaccia ISet <T>.

Descrizione

La classe CSortedSet<T> è un'implementazione del set dinamico di tipo T ordinato, con l'unicità richiesta, di ogni valore. Questa classe fornisce metodi di base per lavorare con set e operazioni correlate, quali: l'unione e l'intersezione di sets, la definizione di sub-sets stretti e non, ed altro.

Dichiarazione

```
template<typename T>
class CSortedSet : public ISet<T>
```

Header

```
#include <Generic\SortedSet.mqh>
```

Inheritance Hierarchy

ICollection
  ISet
    CSortedSet

Class Methods

Metodo	Descrizione
Add	Aggiunge un elemento ad un set insieme
Count	Restituisce il numero di elementi in un set ordinato
Contains	Determina se un set ordinato contiene un elemento con il valore specificato
Comparer	Restituisce un puntatore all'interfaccia IComparer<T>, utilizzata per organizzare un set ordinato
TryGetMin	Ottiene l'elemento minimo da un set ordinato
TryGetMax	Ottiene l'elemento massimo da un set ordinato
CopyTo	Copia tutti gli elementi di un gruppo ordinato all'array specificato partendo dall'indice specificato
Clear	Rimuove tutti gli elementi da un set ordinato
Remove	Rimuove il verificarsi dell'elemento specificato da un set ordinato
ExceptWith	Produce l'operazione di differenza tra la raccolta corrente e una raccolta passata (array)
IntersectWith	Produce il funzionamento dell'intersezione della raccolta corrente ed una raccolta passata (array)

Metodo	Descrizione
SymmetricExceptWith	Produce l'operazione di differenza simmetrica tra la raccolta corrente ed una raccolta passata (array)
UnionWith	Produce l'unione della raccolta corrente ed una raccolta passata (array)
IsProperSubsetOf	Determina se il set ordinato corrente è un subset appropriato della raccolta o dell'array specificati
IsProperSupersetOf	Determina se il set ordinato corrente è un superset appropriato della raccolta o dell'array specificati
IsSubsetOf	Determina se il set ordinato corrente è un subset della raccolta o dell'array specificati
IsSupersetOf	Determina se il set ordinato corrente è un superset della raccolta o dell'array specificati
Overlaps	Determina se il set ordinato corrente si sovrappone alla raccolta o all'array specificati
SetEquals	Determina se il set ordinato corrente contiene tutti gli elementi della raccolta o dell'array specificati
GetViewBetween	Ottiene dal set corrente ordinato un subset specificato dai valori minimi e massimi
GetReverse	Ottiene una copia del set corrente ordinato, in cui tutti gli elementi sono disposti in ordine inverso

Add

Aggiunge un elemento a un set ordinato.

```
bool Add(
   T  value     // il valore dell'elemento
   );
```

Parametri

value

[in] Il valore dell'elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi nel set ordinato.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Contains

Determina se il set ordinato contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore di ricerca
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nel set, altrimenti false.

Comparer

Restituisce un puntatore all'interfaccia IComparer<T>, utilizzata per organizzare un set ordinato.

```
IComparer<T>* Comparer() const;
```

Valore di ritorno

Restituisce un puntatore all'interfaccia IComparer<T>.

TryGetMin

Ottiene l'elemento minimo dall'insieme ordinato.

```
bool TryGetMin(
   T&  min     // la variabile per scrivere il valore
   );
```

Parametri

&min

[out] La variabile in cui verrà scritto il valore minimo.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

TryGetMax

Ottiene l'elemento massimo dall'insieme ordinato.

```
bool TryGetMax(
   T&  max     // una variabile per scrivere il valore
   );
```

Parametri

&max

[out] La variabile in cui verrà scritto il valore massimo.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CopyTo

Copia tutti gli elementi di un set ordinato all'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // l'indice d'inizio per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array nel quale verranno scritti gli elementi del set.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

Clear

Rimuove tutti gli elementi da un set ordinato.

```
void Clear();
```

Remove

Rimuove il verificarsi dell'elemento specificato dal set ordinato.

```
bool Remove(
   T  item     // il valore dell'elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

ExceptWith

Produce l'operazione di differenza tra la raccolta corrente es una raccolta passata (array). Rimuove dall'insieme corrente (array) tutti gli elementi presenti nella raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void ExceptWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta da escludere dall'attuale set ordinato.

&collection[]

[in] Un array da escludere dal set corrente ordinato.

Note

Il risultato viene scritto nella raccolta corrente (array).

IntersectWith

Produce l'operazione dell'intersezione della raccolta corrente e di una raccolta passata (array). Modifica la raccolta corrente per contenere solo elementi presenti nella raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void IntersectWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta con cui verrà intersecato il set corrente.

&collection[]

[in] Un array con cui verrà intersecato il set corrente.

Note

Il risultato viene scritto nella raccolta corrente (array).

SymmetricExceptWith

Produce l'operazione di differenza simmetrica tra la raccolta corrente ed una raccolta passata (array). Modifica la raccolta corrente per contenere solo elementi presenti nell'oggetto sorgente o nella raccolta specificata (array), ma non in entrambi essi.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void SymmetricExceptWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] Una raccolta con cui produrre la differenza simmetrica .

&collection[]

[in] Un array con cui produrre la differenza simmetrica.

Note

Il risultato viene scritto nella raccolta corrente (array).

UnionWith

Produce l'unione della raccolta corrente ed una raccolta passata (array). Aggiunge agli elementi mancanti della raccolta corrente (array) dalla raccolta specificata (array).

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // raccolta
   );
```

Una versione per lavorare con un array.

```
void UnionWith(
   T&  array[]                     // array
   );
```

Parametri

**collection*

[in] A collection with which the current set will be united.

&collection[]

[in] Un array con cui il set corrente verrà unito.

Note

Il risultato viene scritto nella raccolta corrente (array).

IsProperSubsetOf

Determina se il set ordinato corrente è un subset appropriato della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsProperSubsetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set ordinato corrente è un subset adeguato, altrimenti false.

IsProperSupersetOf

Determina se il set ordinato corrente è un superset appropriato della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsProperSupersetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set ordinato corrente è un superset adeguato, altrimenti false.

IsSubsetOf

Determina se il set ordinato corrente è un subset della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsSubsetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set ordinato corrente è un subset, altrimenti false.

IsSupersetOf

Determina se il set ordinato corrente è un superset della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // una raccolta per determinare la relazione
   );
```

Una versione per lavorare con un array.

```
bool IsSupersetOf(
   T&  array[]                     // un array per determinare la relazione
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la relazione.

&collection[]

[in] Un array per determinare la relazione.

Valore di ritorno

Restituisce true se il set ordinato corrente è un superset, altrimenti false.

Overlaps

Determina se il set ordinato corrente si sovrappone alla raccolta o all'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // la raccolta da confrontare
   );
```

Una versione per lavorare con un array.

```
bool Overlaps(
   T&  array[]                     // l'array da confrontare
   );
```

Parametri

**collection*

[in] Una raccolta per determinare la sovrapposizione.

&collection[]

[in] Un array per determinare la sovrapposizione.

Valore di ritorno

Restituisce true se il set corrente ordinato ed una raccolta o un array, si sovrappongono, altrimenti false.

SetEquals

Determina se il set ordinato corrente contiene tutti gli elementi della raccolta o dell'array specificati.

Una versione per lavorare con la raccolta che implementa l'interfaccia ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // la raccolta da confrontare
   );
```

Una versione per lavorare con un array.

```
bool SetEquals(
   T&  array[]                     // l'array da confrontare
   );
```

Parametri

**collection*

[in] Una raccolta per confrontare elementi.

&collection[]

[in] Un array per confrontare gli elementi.

Valore di ritorno

Restituisce true se il set corrente ordinato contiene tutti gli elementi della raccolta o dell'array specificati, altrimenti false.

GetViewBetween

Ottiene dal set corrente ordinato un subset specificato dai valori minimi e massimi.

```
bool GetViewBetween(
   T&  array[],          // un array per la scrittura
   T   lower_value,      // il valore minimo
   T   upper_value       // il valore massimo
   );
```

Parametri

&array[]

[out] Un array per la scrittura del subset.

lower_value

[in] Il valore minimo del range.

upper_value

[in] Il valore massimo del range.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

GetReverse

Ottiene una copia del set corrente ordinato, in cui tutti gli elementi sono disposti in ordine inverso.

```
bool GetReverse(
   T&  array[]     // un array per la scrittura
   );
```

Parametri

&array[]

[out] Un array per la scrittura.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

CStack<T>

CStack<T> è una classe generica che implementa l'interfaccia ICollection<T>.

Descrizione

La classe CStack<T> è una raccolta dinamica di dati tipo T, organizzata come una pila che opera sul principio LIFO (l'ultimo ad entrare è il primo ad uscire).

Dichiarazione

```
template<typename T>
class CStack : public ICollection<T>
```

Header

```
#include <Generic\Stack.mqh>
```

Inheritance Hierarchy

ICollection

CStack

Class Methods

Metodo	Descrizione
Add	Aggiunge un elemento ad una pila
Count	Restituisce il numero di elementi in una pila
Contains	Determina se una pila contiene un elemento con il valore specificato
TrimExcess	Imposta la capacità di una pila al numero effettivo di elementi
CopyTo	Copia tutti gli elementi di una pila nell'array specificato partendo dall'indice specificato
Clear	Rimuove tutti gli elementi da una pila
Remove	Rimuove la prima occorrenza dell'elemento specificato da una pila
Push	Aggiunge un elemento ad una pila
Peek	Restituisce l'elemento di testa senza rimuoverlo da una pila
Pop	Restituisce l'elemento di testa e lo rimuove dalla pila

Add

Aggiunge un elemento allo stack(pila).

```
bool Add(
   T  value     // il valore dell'elemento
   );
```

Parametri

value

[in] Il valore dell'elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Count

Restituisce il numero di elementi nello stack.

```
int Count();
```

Valore di ritorno

Restituisce il numero di elementi.

Contains

Determina se una pila contiene un elemento con il valore specificato.

```
bool Contains(
   T  item     // il valore cercato
   );
```

Parametri

item

[in] Il valore cercato.

Valore di ritorno

Restituisce true se un elemento con il valore specificato si trova nella pila, altrimenti false.

TrimExcess

Imposta la capacità della pila al numero effettivo di elementi e quindi libera la memoria inutilizzata.

```
void TrimExcess();
```

CopyTo

Copia tutti gli elementi di una pila nell'array specificato partendo dall'indice specificato.

```
int CopyTo(
   T&          dst_array[],     // un array per la scrittura
   const int   dst_start=0      // l'indice d'inizio per la scrittura
   );
```

Parametri

&dst_array[]

[out] Un array in cui verranno scritti gli elementi della pila.

dst_start=0

[in] Un indice nell'array da cui inizia la copia.

Valore di ritorno

Restituisce il numero di elementi copiati.

Clear

Rimuove tutti gli elementi da una pila(Stack).

```
void Clear();
```

Remove

Rimuove la prima occorrenza dell'elemento specificato dalla pila.

```
bool Remove(
   T  item     // il valore dell'elemento
   );
```

Parametri

item

[in] Il valore dell'elemento da eliminare.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Push

Aggiunge un elemento allo stack(pila).

```
bool Push(
   T  value     // elemento da aggiungere
   );
```

Parametri

value

[in] Un elemento da aggiungere.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Peek

Restituisce l'elemento di testa senza rimuoverlo dalla pila.

```
T Peek();
```

Valore di ritorno

Restituisce l'elemento testa.

Pop

Restituisce l'elemento di testa e lo rimuove dalla pila.

```
T Pop();
```

Valore di ritorno

Restituisce l'elemento testa.

ArrayBinarySearch

Cerca il valore specificato in un array unidimensionale ordinato in ordine crescente utilizzando l'interfaccia IComparable<T> per confrontare gli elementi.

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],          // un array per la ricerca
   const int       start_index,      // l'indice per l'inizio
   const int       count,            // il raggio di ricerca
   T               value,            // il valore di ricerca
   IComparer<T>*   comparer          // interfaccia da confrontare
   );
```

Parametri

&array[]

[out] L'array in cui cercare.

value

[in] Il valore cercato.

**comparer*

[in] Un interfaccia per confrontare elementi.

start_index

[in] L'indice d'inizio dal quale inizia la ricerca.

count

[in] La lunghezza del range di ricerca.

Valore di ritorno

Restituisce l'indice dell'elemento trovato. Se il valore di ricerca non viene trovato, restituisce l'indice dell'elemento più piccolo, che è il valore più vicino.

ArrayIndexOf

Cerca la prima occorrenza di un valore in un array unidimensionale.

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          // un array per la ricerca
   T           value,            // il valore ricercato
   const int   start_index,      // l'indice d'inizio
   const int   count             // il raggio di ricerca
   );
```

Parametri

&array[]

[out] L'array in cui cercare.

value

[in] Il valore cercato.

start_index

[in] L'indice d'inizio dal quale inizia la ricerca.

count

[in] La lunghezza del range di ricerca.

Valore di ritorno

Restituisce l'indice del primo elemento trovato. Se il valore non viene trovato, restituisce -1.

ArrayLastIndexOf

Cerca l'ultima occorrenza di un valore in un'array unidimensionale.

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          // un array per la ricerca
   T           value,            // il valore ricercato
   const int   start_index,      // l'indice iniziale
   const int   count             // il raggio di ricerca
   );
```

Parametri

&array[]

[out] L'array in cui cercare.

value

[in] Il valore cercato.

start_index

[in] L'indice d'inizio dal quale inizia la ricerca.

count

[in] La lunghezza del range di ricerca.

Valore di ritorno

Restituisce l'indice dell'ultimo elemento trovato. Se il valore non viene trovato, restituisce -1.

ArrayReverse

Modifica la sequenza di elementi in un array unidimensionale.

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // l'array sorgente
   const int   start_index,      // l'indice iniziale
   const int   count             // il numero di elementi
   );
```

Parametri

&array[]

[out] L'array sorgente.

start_index

[in] L'indice d'inizio.

count

[in] Il numero di elementi dell' array che partecipano all'operazione.

Valore di ritorno

Restituisce true in caso di successo, altrimenti false.

Compare

Confronta i due valori, se uno di loro è "maggiore, minore o uguale" ad un altro.

Una versione per confrontare due valori bool.

```
int Compare(
   const bool  x,          // il primo valore
   const bool  y           // il secondo valore
   );
```

Una versione per confrontare due valori char.

```
int Compare(
   const char  x,          // il primo valore
   const char  y           // il secondo valore
   );
```

Una versione per confrontare due valori uchar.

```
int Compare(
   const uchar  x,         // il primo valore
   const uchar  y          // il secondo valore
   );
```

Una versione per confrontare due valori short.

```
int Compare(
   const short  x,         // il primo valore
   const short  y          // il secondo valore
   );
```

Una versione per confrontare due valori di ushort.

```
int Compare(
   const ushort  x,        // il primo valore
   const ushort  y         // il secondo valore
   );
```

Una versione per confrontare due valori color.

```
int Compare(
   const color  x,         // il primo valore
   const color  y          // il secondo valore
   );
```

Una versione per confrontare due valori int.

```
int Compare(
   const int  x,           // il primo valore
   const int  y            // il secondo valore
   );
```

Una versione per confrontare due valori uint.

```
int Compare(
   const uint  x,          // il primo valore
   const uint  y           // il secondo valore
   );
```

Una versione per il confronto di due valori datetime.

```
int Compare(
   const datetime  x,      // il primo valore
   const datetime  y       // il secondo valore
   );
```

Una versione per confrontare due valori long.

```
int Compare(
   const long  x,          // il primo valore
   const long  y           // il secondo valore
   );
```

Una versione per confrontare due valori ulong.

```
int Compare(
   const ulong  x,         // il primo valore
   const ulong  y          // il secondo valore
   );
```

Una versione per confrontare due valori float.

```
int Compare(
   const float  x,         // il primo valore
   const float  y          // il secondo valore
   );
```

Una versione per confrontare due valori double.

```
int Compare(
   const double  x,        // il primo valore
   const double  y         // il secondo valore
   );
```

Una versione per confrontare due valori stringa.

```
int Compare(
   const string  x,        // il primo valore
   const string  y         // il secondo valore
   );
```

Una versione per confrontare due valori di altri tipi.

```
template<typename T>
int Compare(
```

```
   T  x,                  // il primo valore
   T  y                   // il secondo valore
   );
```

Parametri

x

[in] Il primo valore

y

[in] Il secondo valore

Valore di ritorno

Restituisce un numero che esprime il rapporto tra i due valori confrontati:

- se il risultato è inferiore a zero, x è inferiore a y (x<y)
- se il risultato è uguale a zero, x è uguale a y (x=y)
- se il risultato è maggiore di zero, x è maggiore di y (x>y)

Note

Se il tipo T è un oggetto che implementa l'interfaccia IComparable<T>, allora gli oggetti verranno confrontati in base al metodo Compare. In tutti gli altri casi viene restituito 0.

Equals

Confronta due valori per l'uguaglianza.

```
template<typename T>
bool Equals(
   T  x,      // il primo valore
   T  y       // il secondo valore
   );
```

Parametri

x

[in] Il primo valore

y

[in] Il secondo valore

Valore di ritorno

Restituisce true se gli oggetti sono uguali, altrimenti false.

Note

Se il tipo T è un oggetto che implementa l'interfaccia IEqualityComparable<T>, allora gli oggetti saranno confrontati in base al metodo di confronto Equals. Il confronto standard per uguaglianza viene utilizzato in tutti gli altri casi.

GetHashCode

Calcola il valore del codice hash.

Una versione per il lavoro con il tipo bool.

```
int GetHashCode(
   const bool   value          // valore
   );
```

Una versione per il lavoro con il tipo char.

```
int GetHashCode(
   const char   value          // valore
   );
```

Una versione per il lavoro con il tipo uchar.

```
int GetHashCode(
   const uchar   value         // valore
   );
```

Una versione per il lavoro con il tipo short.

```
int GetHashCode(
   const short   value         // valore
   );
```

Una versione per il lavoro con il tipo ushort.

```
int GetHashCode(
   const ushort   value        // valore
   );
```

Una versione per il lavoro con il tipo color

```
int GetHashCode(
   const color   value         // valore
   );
```

Una versione per il lavoro con il tipo int.

```
int GetHashCode(
   const int   value           // valore
   );
```

Una versione per il lavoro con il tipo uint.

```
int GetHashCode(
   const uint   value          // valore
   );
```

Una versione per il lavoro con il tipo datetime.

```
int GetHashCode(
```

```
   const datetime   value     // valore
   );
```

Una versione per il lavoro con il tipo long.

```
int GetHashCode(
   const long   value         // valore
   );
```

Una versione per il lavoro con il tipo ulong.

```
int GetHashCode(
   const ulong   value        // valore
   );
```

Una versione per il lavoro con il tipo float.

```
int GetHashCode(
   const float   value        // valore
   );
```

Una versione per il lavoro con il tipo double.

```
int GetHashCode(
   const double   value       // valore
   );
```

Una versione per il lavoro con il tipo string.

```
int GetHashCode(
   const string   value       // valore
   );
```

Una versione per il lavoro con altri tipi.

```
template<typename T>
int GetHashCode(
   T   value                  // valore
   );
```

Parametri

value

[in] Il valore per cui si desidera ottenere il codice hash.

Valore di ritorno

Restituisce il codice hash.

Note

Se il tipo T è un oggetto che implementa l'interfaccia IEqualityComparable<T>, allora il codice hash sarà ottenuto in base al suo metodo HashCode. In tutti gli altri casi, il codice hash verrà calcolato come valore hash del nome del tipo di valore.

Operazioni sui File

Questa sezione contiene i dettagli tecnici delle classi operazioni sui file e le descrizioni dei corrispondenti componenti della libreria standard MQL5.

Le classi operazioni sui file faranno risparmiare tempo nello sviluppo di applicazioni utilizzando le operazioni di input/output.

La MQL5 standard Library (in termini di operazioni di file) si trova nella directory di lavoro del terminale nella cartella Include\Files.

Classe	Descrizione
CFile	Clacce base di operazioni sui file
CFileBin	Classe di operazione sui file binari
CFileTxt	Classe di operazione sui file di testo

CFile

CFile è una classe base per le classi CFileBin e CFileTxt.

Descrizione

La classe CFile fornisce l'accesso semplificato ai file e cartelle di funzioni di API MQL5 per tutti i suoi discendenti.

Dichiarazione

```
class CFile: public CObject
```

Titolo

```
#include <Files\File.mqh>
```

Gerarchia di ereditarietà

CObject

CFile

Discendenti diretti

CFileBin, CFilePipe, CFileTxt

I Metodi della Classe per Gruppi

Attributi	
Handle	Ottiene handle del file
Filename	Ottiene il nome del file
Flags	Ottiene le flag del file
SetUnicode	Imposta/Cancella la flag FILE_UNICODE
SetCommon	Imposta/Cancella la flag FILE_COMMON
Metodi generali per i file	
Open	Apre il file
Close	Chiude il file
Delete	Elimina il file
IsExist	Controlla l'esistenza del file
Copy	Copia il file
Move	Rinomina/sposta il file
Size	Ottiene la grandezza del file
Tell	Ottiene la posizione del file corrente

Attributi	
Seek	Imposta la posizione del file corrente
Flush	Flusha i dati sul disco
IsEnding	Controlla la fine del file
IsLineEnding	Controlla la riga per la file
Metodi generali per le cartelle	
FolderCreate	Crea cartella
FolderDelete	Elimina cartella
FolderClean	Pulisce cartella
I metodi di ricerca di file e cartelle	
FileFindFirst	Inizia la ricerca di file
FileFindNext	Continua la ricerca di file
FileFindClose	Chiude handle di ricerca

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Handle

Ottiene il file handle del file aperto.

```
int  Handle()
```

Valore di ritorno

Handle del file aperto assegnato all' istanza della classe. Se non ci sono file assegnati, restituisce -1.

FileName

Ottiene il nome del file del file aperto.

```
string  FileName()
```

Valore di ritorno

Il nome del file aperto, assegnato all'istanza della classe. Se non ci sono file assegnati, esso restituisce "".

Flags

Ottiene flags del file aperto.

```
int  Flags()
```

Valore di ritorno

Flags del file aperto assegnato alla istanza della classe.

SetUnicode

Imposta/Cancella la flag FILE_UNICODE.

```
void  SetUnicode(
   bool  unicode      // valore flag
   )
```

Parametri

unicode

[in] Nuovo valore per la flag FILE_UNICODE.

Nota

Il risultato di operazioni sulle stringhe dipende dalla flag FILE_UNICODE. Se è falso, i codici ANSI vengono utilizzati (simboli un byte). Se è impostato, i codici ANSI vengono utilizzati (simboli due byte). Se il file è già aperto, la flag non può essere modificata.

SetCommon

Imposta/Cancella la flag FILE_COMMON.

```
void  SetCommon(
   bool  common      // valore flag
   )
```

Parametri

common

[in] Nuovo valore per la flag FILE_COMMON.

Nota

La flag FILE_COMMON determina la cartella di lavoro corrente. Se è falsa, la cartella del terminale locale viene utilizzata come cartella di lavoro corrente. Se è true, la cartella dati comune è utilizzata come cartella di lavoro corrente. Se il file è già aperto, la flag non può essere modificata.

Open

Apre il file specificato e, in caso di successo, lo assegna all'istanza della classe.

```
int  Open(
   const string  file_name,        // nome del file
   int           flags,            // flags
   short         delimiter=9       // separatore
   )
```

Parametri

file_name

[in] Nome del File da aprire.

flags

[in] Flags del file da aprire.

delimiter=9

[in] CSV separatore file.

Valore di ritorno

Handle del file aperto.

Nota

La cartella di lavoro dipende dalla flag che è stata precedentemente impostata/resettata utilizzando il metoto SetCommon().

Close

Chiude il file assegnato alla istanza della classe.

```
void  Close()
```

Delete

Elimina il file assegnato all'istanza del file.

```
void  Delete()
```

Delete

Elimina il file specificato.

```
void  Delete(
   const string  file_name      // nome del file
   )
```

Parametri

file_name

[in] Nome del file da eliminare.

Nota

La cartella di lavoro dipende dal flag che è stato precedentemente impostato/resettato utilizzando il metodo SetCommon().

IsExist

Controlla l'esistenza del file

```
bool  IsExist(
   const string  file_name      // nome del file
   )
```

Parametri

file_name

[in] Nome del file da controllare.

Valore di ritorno

true - il file esiste.

Copy

Copia un file.

```
bool  Copy(
   const string  src_name,      // nome del file
   int           src_flag,      // flag
   const string  dst_name,      // nome del file
   int           dst_flags      // flags
   )
```

Parametri

src_name

[in] Nome del file sorgente.

src_flag

[in] Flags del file sorgente (solo FILE_COMMON viene usato).

dst_name

[in] Nome file, del file di destinazione.

dst_flags

[in] Flags del file di destinazione (solo FILE_REWRITE e FILE_COMMON sono usati).

Valore di ritorno

true - successo, false - non può copiare il file.

Move

Rinomina/sposta file.

```
bool  Move(
   const string  src_name,      // nome del file
   int           src_flag,      // flag
   const string  dst_name,      // nome del file
   int           dst_flags      // flags
   )
```

Parametri

src_name

[in] File name sorgente.

src_flag

[in] Flags del file sorgente (solo FILE_COMMON viene usato).

dst_name

[in] Nome file, del file di destinazione.

dst_flags

[in] Flags del file di destinazione (solo FILE_REWRITE e FILE_COMMON sono usati).

Valore di ritorno

true - successo, false - non è riuscito a spostare/rinominare il file.

Size

Ottiene la dimensione del file in byte.

```
ulong  Size()
```

Valore di ritorno

dimensione del file in byte. Se non ci sono file assegnati, restituisce ULONG_MAX.

Tell

Ottiene la posizione del puntatore del file corrente.

```
ulong  Tell()
```

Valore di ritorno

la posizione del file corrente. Se non ci sono file assegnati, restituisce ULONG_MAX.

Seek

Imposta la posizione del puntatore del file.

```
void  Seek(
   long                offset,     // offset
   ENUM_FILE_POSITION  origin      // origine
   )
```

Parametri

offset

[in] Offset del file in bytes (può essere negativo).

origin

[in] Origine dell'offset.

Valore di ritorno

true - successo, false - non posso cambiare il puntatore del file.

Flush

Svuota tutti i dati buffer di input/output su disco.

```
void  Flush()
```

IsEnding

Controlla la fine dei file durante le operezioni di lettura file.

```
bool  IsEnding()
```

Valore di ritorno

true - la fine del file è stata raggiunta durante l'operazione di lettura o ricerca.

IsLineEnding

Controlla la fine di un file di testo durante l'operazione di lettura del file.

```
bool  IsLineEnding()
```

Valore di ritorno

true - la fine della linea è stata raggiunta nel corso della lettura di di un file txt o CSV (caratteri CR-LF).

FolderCreate

Crea nuova cartella.

```
bool  FolderCreate(
   const string  folder_name      // nome cartella
   )
```

Parametri

folder_name

[in] Nome della cartella da creare. Contiene il percorso della cartella relativa alla cartella definita dalla flag FILE_COMMON.

Valore di ritorno

true - successo, false - non si può creare la cartella.

Nota

La cartella di lavoro dipende dal flag che è stato precedentemente impostato/resettato utilizzando il metodo SetCommon().

FolderDelete

Elimina cartella specificata.

```
bool  FolderDelete(
   const string  folder_name      // nome cartella
   )
```

Parametri

folder_name

[in] Nome della cartella da eliminare. Contiene il percorso della cartella relativa alla cartella definita dalla flag FILE_COMMON.

Valore di ritorno

true - successo, false - non si può eliminare l'ordine.

Nota

La cartella di lavoro dipende dal flag che è stato precedentemente impostato/resettato utilizzando il metodo SetCommon().

FolderClean

Pulisce la cartella specificata.

```
bool  FolderClean(
   const string  folder_name      // nome cartella
   )
```

Parametri

folder_name

[in] Nome della cartella da ripulire. Contiene il percorso della cartella relativa alla cartella definita dalla flag FILE_COMMON.

Valore di ritorno

true - successo, e false - non si può cambiare la cartella.

Nota

La cartella di lavoro dipende dal flag precedentemente impostato/resettato dal metodo SetCommon().

FileFindFirst

Inizia la ricerca di file utilizzando il filtro specificato.

```
int  FileFindFirst(
   const string  filter,        // filtro di ricerca
   string&       file_name      // riferimento
   )
```

Parametri

filter

[in] Filtro di ricerca.

file_name

[out] Il riferimento al nome file della stringa del primo file trovato è inserito nel in caso di successo.

Valore di ritorno

L' handle che può essere utilizzato per ulteriore ricerca di file utilizzando FileFindNext; INVALID_HANDLE se non ci sono i file che corrispondono al filtro.

Nota

La cartella di lavoro dipende dal flag precedentemente impostato/resettato dal metodo SetCommon().

FileFindNext

Continua la ricerca di file avviata dal metodo FileFindFirst().

```
bool  FileFindNext(
   int      search_handle,     // handle di ricerca
   string&  file_name          // riferimento
   )
```

Parametri

search_handle

[in] Handle di ricerca restituito dal metodo FileFindFirst().

file_name

[in] Il riferimento alla stringa; il nome del file trovato viene inserito in caso di successo.

Valore di ritorno

true - in caso di successo, false - non ci sono file che corrispondono al filtro.

FileFindClose

Chiude handle di ricerca.

```
void  FileFindClose(
   int  search_handle      // handle di ricerca
   )
```

Parametri

search_handle

[in] Handle di ricerca restituito dal metodo FileFindFirst().

CFileBin

CFileBin è una classe per l'accesso semplificato ai file binari.

Descrizione

La Classe CFileBin fornisce l'accesso ai file binari.

Dichiarazione

```
class CFileBin: public CFile
```

Titolo

```
#include <Files\FileBin.mqh>
```

Gerarchia di ereditarietà

CObject
  CFile
    CFileBin

I Metodi della Classe per Gruppi

Metodi Open	
Open	Apre un file binario
Metodi Write	
WriteChar	Scrive variabili di tipo char o uchar
WriteShort	Scrive variabili di tipo short o ushort
WriteInteger	Scrive variabili di tipo int o uint
WriteLong	Scrive variabili di tipo long o ulong
WriteFloat	Scrive variabili di tipo float
WriteDouble	Scrive variabili di tipo double
WriteString	Scrive variabili di tipo string
WriteCharArray	Scrive un array di variabili di tipo char o uchar
WriteShortArray	Scrive un array di variabili di tipo short o ushort
WriteIntegerArray	Scrive un array di variabili di tipo int o uint
WriteLongArray	Scrive un array di variabili di tipo long o ulong
WriteFloatArray	Scrive array di variabili float
WriteDoubleArray	Scrive un array di variabili di tipo double
WriteObject	Scrive dati dell'istanza della classe erede CObject

Metodi Open	
Metodi Read	
ReadChar	Legge variabili di tipo char o uchar
ReadShort	Legge variabili di tipo short o ushort
ReadInteger	Legge int or uint type variable
ReadLong	Legge variabili di tipo long o ulong
ReadFloat	Legge variabili di tipo float
ReadDouble	Legge variabili di tipo double
ReadString	Legge variabili di tipo string
ReadCharArray	Legge un array di variabili di tipo char o uchar
ReadShortArray	Legge un array di variabili di tipo short o ushort
ReadIntegerArray	Legge un array di variabili di tipo int o uint
ReadLongArray	Legge un array di variabili di tipo long o ulong
ReadFloatArray	Legge un array di variabili di tipo float
ReadDoubleArray	Legge un array di variabili di tipo double
ReadObject	Legge dati dell'istanza della classe erede CObject

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

Open

Apre il file binario specificato e, in caso di successo, assegna l'istanza della classe.

```
int  Open(
   const string  file_name,      // nome file
   int           flags           // flags
   )
```

Parametri

file_name

[in] nome del file da aprire.

flags

[in] Flags di apertura file (la flag FILE_BIN è impostata forzatamente).

Valore di ritorno

Handle del file aperto.

WriteChar

Scrive variabile di tipo char o uchar da file.

```
uint  WriteChar(
   char  value      // valore
   )
```

Parametri

value

[in] Variabile da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteShort

Scrive variabile di tipo short o ushort in un file.

```
uint  WriteShort(
   short  value      // valore
   )
```

Parametri

value

[in] Variabile da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteInteger

Scrive tipo di variabile int o uint da file.

```
uint  WriteInteger(
   int  value      // valore
   )
```

Parametri

value

[in] Variabile da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteLong

Scrive tipo di variabile long o ulong in un file.

```
uint  WriteLong(
   long  value      // valore
   )
```

Parametri

value

[in] Variabile da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteFloat

Scrive variabili di tipo float in un file.

```
uint  WriteFloat(
   float  value      // valore
   )
```

Parametri

value

[in] Variabile da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteDouble

Scrive variabili di tipo double in un file.

```
uint  WriteDouble(
   double  value      // valore
   )
```

Parametri

value

[in] Variabile da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteString

Scrive variabili di tipo string in un file.

```
uint  WriteString(
   const string  value      // valore
   )
```

Parametri

value

[in] Stringa da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteString

Scrive variabili di tipo string in un file.

```
uint  WriteString(
   const string  value,     // valore
   int           size       // grandezza
   )
```

Parametri

value

[in] Stringa da scrivere.

size

[in] Numero di byte da scrivere.

Valore di ritorno

Numero di byte scritti.

WriteCharArray

Scrive un array di variabili di tipo char o uchat su file.

```
uint  WriteCharArray(
   char&  array[],            // array
   int    start_item=0,       // elemento iniziale
   int    items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Array da scrivere.

start_item=0

[in] Elemento da cui iniziare a scrivere.

items_count=-1

[in] Numero di elementi da scrivere (-1 - intero array).

Valore di ritorno

Numero di byte scritti.

WriteShortArray

Scrive un array di variabili di tipo short o ushort in un file.

```
uint  WriteShortArray(
   short&  array[],            // array
   int     start_item=0,       // elemento iniziale
   int     items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Array da scrivere.

start_item=0

[in] Elemento da cui iniziare a scrivere.

items_count=-1

[in] Numero di elementi da scrivere (-1 - intero array).

Valore di ritorno

Numero di byte scritti.

WriteIntegerArray

Scrive un array di variabili di tipo int o uint in un file.

```
uint  WriteIntegerArray(
   int&  array[],             // array
   int   start_item=0,        // elemento inziale
   int   items_count=-1       // numero di elementi
   )
```

Parametri

array[]

[in] Array da scrivere.

start_item=0

[in] Elemento da cui iniziare a scrivere.

items_count=-1

[in] Numero di elementi da scrivere (-1 - intero array).

Valore di ritorno

Numero di byte scritti.

WriteLongArray

Scrive un array di variabili di tipo long o ulong.

```
uint  WriteLongArray(
   long&  array[],            // array
   int    start_item=0,       // elemento iniziale
   int    items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Array da scrivere.

start_item=0

[in] Elemento da cui iniziare a scrivere.

items_count=-1

[in] Numero di elementi da scrivere (-1 - intero array).

Valore di ritorno

Numero di byte scritti.

WriteFloatArray

Scrive una serie di variabili di tipo array in un file.

```
uint  WriteFloatArray(
   float&  array[],            // array
   int     start_item=0,       // elemento iniziale
   int     items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Array da scrivere.

start_item=0

[in] Elemento da cui iniziare a scrivere.

items_count=-1

[in] Numero di elementi da scrivere (-1 - intero array).

Valore di ritorno

Numero di byte scritti.

WriteDoubleArray

Scrive un array di variabili di tipo double in un file.

```
uint  WriteDoubleArray(
   double&  array[],             // array da scrivere
   int      start_item=0,        // elemento iniziale
   int      items_count=-1       // numero di elementi
   )
```

Parametri

array[]

[in] Array da scrivere.

start_item=0

[in] Elemento da cui iniziare a scrivere.

items_count=-1

[in] Numero di elementi da scrivere (-1 - intero array).

Valore di ritorno

Numero di byte scritti.

WriteObject

Scrive dati dell'istanza della classe erede CObject in un file

```
bool  WriteObject(
   CObject*  object      // riferimento alla variabile
   )
```

Parametri

object

[in] Riferimento alla istanza della classe erede CObject da scrivere.

Valore di ritorno

true - successo, false - non posso scrivere i dati.

ReadChar

Legge variabile di tipo char o uchar da file.

```
bool  ReadChar(
   char&  value      // valore flag
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadShort

Legge tipo di variabile short o ushort da file.

```
bool  ReadShort(
   short&  value
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadInteger

Legge tipo di variabile int o uint da file.

```
bool  ReadInteger(
   int&  value      // variabile
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadLong

Legge tipo di variabile long o ulong da file.

```
bool  ReadLong(
   long&  value
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadFloat

Legge il tipo di variabile float da file.

```
bool  ReadFloat(
   float&  value      // variabile
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadDouble

Legge il tipo di variabile double da file.

```
bool  ReadDouble(
   double&  value
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadString

Legge variabile di tipo string dal file.

```
bool  ReadString(
   string&  value      // stringa
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadString

Legge variabile di tipo string dal file.

```
bool  ReadString(
   string&  value
   )
```

Parametri

value

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadCharArray

Legge un array di variabili di tipo char o uchar da file.

```
bool  ReadCharArray(
   char&  array[],            // array
   int    start_item=0,       // elemento iniziale
   int    items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

start_item=0

[in] Elemento iniziale da cui leggere.

items_count=-1

[in] Numero di elementi da leggere (-1 - legge fino alla fine del file).

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadShortArray

Legge un array di variabili di tipo short o ushort da file.

```
bool  ReadShortArray(
   short&  array[],            // array
   int     start_item=0,       // elemento iniziale
   int     items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

start_item=0

[in] Elemento iniziale da cui leggere.

items_count=-1

[in] Numero di elementi da leggere (-1 - legge fino alla fine del file).

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadIntegerArray

Legge un array di variabili di tipo int o uint da file.

```
bool  ReadIntegerArray(
   int&  array[],             // array
   int   start_item=0,        // elemento inziale
   int   items_count=-1       // numero di elementi
   )
```

Parametri

array[]

[in] Riferimento all' array target di tipo int o uint.

start_item=0

[in] Elemento iniziale da cui leggere.

items_count=-1

[in] Numero di elementi da leggere (-1 - legge fino alla fine del file).

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadLongArray

Legge un array di variabili di tipo long o ulong da file.

```
bool  ReadLongArray(
   long&  array[],            // array
   int    start_item=0,       // elemento iniziale
   int    items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

start_item=0

[in] Elemento iniziale da cui leggere.

items_count=-1

[in] Numero di elementi da leggere (-1 - legge fino alla fine del file).

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadFloatArray

Legge un array di variabili di tipo float da file.

```
bool  ReadFloatArray(
   float&  array[],            // array
   int     start_item=0,       // elemento iniziale
   int     items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

start_item=0

[in] Elemento iniziale da cui leggere.

items_count=-1

[in] Numero di elementi da leggere (-1 - legge fino alla fine del file).

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadDoubleArray

Legge un array di variabili di tipo double da file.

```
bool  ReadDoubleArray(
   double&  array[],            // array
   int      start_item=0,       // elemento iniziale
   int      items_count=-1      // numero di elementi
   )
```

Parametri

array[]

[in] Riferimento alla variabile per l'immissione dei dati di lettura.

start_item=0

[in] Elemento iniziale da cui leggere.

items_count=-1

[in] Numero di elementi da leggere (-1 - legge fino alla fine del file).

Valore di ritorno

true - successo, false - non può leggere i dati.

ReadObject

Legge dati dell'istanza della classe erede CObject da file

```
bool  ReadObject(
   CObject*  object      // puntatore
   )
```

Parametri

object

[in] Puntatore all'istanza della classe erede CObject da leggere.

Valore di ritorno

true - successo, false - non può leggere i dati.

CFileTxt

CFileTxt è una classe per l'accesso semplificato ai file di testo.

Descrizione

La Classe CFileTxt fornisce l'accesso ai file di testo.

Dichiarazione

```
class CFileTxt: public CFile
```

Titolo

```
#include <Files\FileTxt.mqh>
```

Gerarchia di ereditarietà

CObject
  CFile
    CFileTxt

I Metodi della Classe per Gruppi

Metodi Open	
Open	Apre un file di testo
Metodi Write	
WriteString	Scrive variabili di tipo string
Metodi Read	
ReadString	Legge variabili di tipo string

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

Open

Apre un file di testo specificato e, in caso di successo, lo assegna all'istanza della classe.

```
int  Open(
   const string  file_name,      // nome file
   int           flags           // flags
   )
```

Parametri

file_name

[in] Nome del File da aprire.

flags

[in] Flag di apertura file (FILE_TXT flag è impostata forzatamente).

Valore di ritorno

Handle di file aperto.

WriteString

Scrive tipo variabile stringa in un file.

```
uint  WriteString(
   const string  value      // stringa
   )
```

Parametri

value

[in] Stringa da scrivere.

Valore di ritorno

Numero di byte scritti.

ReadString

Legge variabile di tipo string dal file.

```
string  ReadString()
```

Valore di ritorno

Stringa che è stata letta.

Operazioni sulle stringhe

Questa sezione contiene i dettagli tecnici delle classi di operazioni stringa e le descrizioni dei corrispondenti componenti della libreria standard MQL5.

L'uso delle classi di operazioni sulle stringhe farà risparmiare tempo nello sviluppo di applicazioni di elaborazione dei dati testuali.

La MQL5 standard library (in termini di operazioni sulle stringhe) si trova nella directory di lavoro del terminale nella cartella Include\Strings.

Classe	Descrizione
CString	Classe per operazioni sulle stringhe

CString

CString è una classe per l'accesso semplificato alle variabili di tipo stringa.

Descrizione

La classe CString fornisce un accesso semplificato alle funzioni API MQL5 che lavorano con variabili stringa.

Dichiarazione

```
class CString: public CObject
```

Titolo

```
#include <Strings\String.mqh>
```

Gerarchia di ereditarietà

CObject
  CString

I Metodi della Classe per Gruppi

Metodi di accesso ai dati	
Str	Ottiene una stringa
Len	Ottiene la lunghezza di una stringa
Copy	Copia una stringa
Fill methods	
Fill	Riempie una stringa
Assign	Assegna una stringa
Append	Aggiunge una stringa
Insert	Inserisce una stringa
Confronta i metodi	
Compare	confronta stringhe
CompareNoCase	Esegue un confronto case insensitive di stringhe
Metodi sottostringa	
Left	Ottiene una sottostringa da sinistra
Right	Ottiene una sottostringa da destra
Mid	Ottiene una sottostringa da metà

Metodi di accesso ai dati	
Metodi Trimma/Elimina	
Trim	Rifila(trimma) una stringa da sinistra e da destra
TrimLeft	Rifila una stringa da sinistra
TrimRight	Rifila una stringa da destra
Clear	Cancella una stringa
Metodi di conversione	
ToUpper	Converte una stringa in maiuscolo.
ToLower	Converte una stringa in minuscolo.
Reverse	Inverte una stringa
Metodi di ricerca	
Find	Cerca una sottostringa da sinistra a destra
FindRev	Cerca una sottostringa da destra a sinistra
Remove	Elimina una sottostringa
Replace	Sostituisce una sottostringa

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type

Str

Ottiene la stringa.

```
string  Str() const;
```

Valore di ritorno

Copia della stringa.

Len

Ottiene la lunghezza della stringa.

```
uint  Len() const;
```

Valore di ritorno

Lunghezza della stringa.

Copy

Copia una stringa per riferimento.

```
void  Copy(
   string&  copy      // riferimento
   ) const;
```

Parametri

copy

[in] Riferimento ad una stringa di dati.

Copy

Copia una stringa per l'istanza della classe CString.

```
void  Copy(
   CString*  copy      // puntatore
   ) const;
```

Parametri

copy

[in] Puntatore ad un'istanza della classe CString per i dati.

Fill

Riempie una stringa con il carattere specificato.

```
bool  Fill(
   short  character      // carattere
   )
```

Parametri

character

[In] Carattere per il riempimento della stringa.

Valore di ritorno

true - successo, false - non può riempire la stringa.

Assign

Assegna una stringa.

```
void  Assign(
   const string  str      // stringa
   )
```

Parametri

str

[in] Stringa da assegnare.

Assign

Assegna una stringa dall'istanza della classe CString.

```
void  Assign(
   CString*  str      // puntatore
   )
```

Parametri

str

[in] Puntatore all'istanza della classe CString da assegnare.

Append

Aggiunge una stringa.

```
void  Append(
   const string  str      // stringa
   )
```

Parametri

str

[in] Stringa da aggiungere.

Append

Aggiunge una stringa dall'istanza della classe CString.

```
void  Append(
   CString*  string      // puntatore
   )
```

Parametri

string

[in] Puntatore all'istanza della classe CString da aggiungere.

Insert

Inserisce una stringa nella posizione specificata.

```
uint  Insert(
   uint          pos,      // posizione
   const string  str       // stringa
   )
```

Parametri

pos

[in] Inserimento posizione.

str

[in] Stringa da inserire.

Valore di ritorno

Lunghezza della stringa risultatante.

Insert

Inserisce una stringa alla posizione specificata dall'della classe CString.

```
uint  Insert(
   uint      pos,      // posizione
   CString*  str       // puntatore
   )
```

Parametri

pos

[in] Posizione in cui inserire.

str

[in] Puntatore all'istanza della classe CString da inserire.

Valore di ritorno

Lunghezza della stringa risultatante.

Compare

Confronta una stringa.

```
int  Compare(
   const string  str      // stringa
   ) const;
```

Parametri

str

[in] Stringa da confrontare.

Valore di ritorno

0 - entrambe le stringhe sono uguali, -1 - la stringa classe è inferiore rispetto alla stringa di confrontare, 1 - la stringa classe è maggiore della stringa per confrontare.

Compare

Confronta un istanza della classe stringa CString.

```
int  Compare(
   CString*  str      // puntatore
   ) const;
```

Parametri

str

[in] Puntatore all'istanza della classe CString da confrontare.

Valore di ritorno

0 - le stringhe sono uguali, -1 - la stringa classe è inferiore rispetto alla stringa di confrontare, 1 - la stringa classe è maggiore della stringa per confrontare.

CompareNoCase

Esegue un confronto case insensitive di stringhe.

```
int  CompareNoCase(
   const string  str      // stringa
   ) const;
```

Parametri

str

[in] Stringa da confrontare.

Valore di ritorno

0 - le stringhe sono uguali, -1 - la stringa classe è inferiore rispetto alla stringa di confrontare, 1 - la stringa classe è maggiore della stringa per confrontare.

CompareNoCase

Confronta una stringa (case insensitive) con una istanza di classe stringa CString.

```
int  CompareNoCase(
   CString*  str      // puntatore
   ) const;
```

Parametri

str

[in] Puntatore all'istanza della classe CString da confrontare.

Valore di ritorno

0 - se le stringhe sono uguali, -1 - la stringa classe è inferiore rispetto alla stringa di confrontare, 1 - la stringa classe è maggiore della stringa per confrontare.

Left

Ottiene la sottostringa di una lunghezza specificata dall'inizio di una stringa.

```
string  Left(
   uint  count      // lunghezza
   )
```

Parametri

count

[in] Lunghezza della sottostringa.

Valore di ritorno

Sottostringa risultante.

Right

Ottiene un sottostringa di una lunghezza specificata dalla fine della stringa.

```
string  Right(
   uint  count      // lunghezza
   )
```

Parametri

count

[in] Lunghezza della sottostringa.

Valore di ritorno

Sottostringa risultante.

Mid

Ottiene la sottostringa di una lunghezza specificata da una posizione stringa specificata.

```
string  Mid(
   uint  pos,        // posizione
   uint  count       // lunghezza
   )
```

Parametri

pos

[in] Posizione sottostringa.

count

[in] Lunghezza della sottostringa.

Valore di ritorno

Sottostringa risultante.

Trim

Rimuove tutti i caratteri all'interno di un insieme (così come ' ','\t','\r','\n') ad entrambe le estremità di una stringa da questa stringa.

```
int  Trim(
   const string  targets      // il set
   )
```

Parametri

targets

[in] Set di caratteri da rimuovere.

Valore di ritorno

Numero di caratteri rimossi.

Esempio:

```
//--- esempio per CString::Trim
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("Stringa sorgente '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("Stringa risultato '%s'",str.Str());
  }
```

TrimLeft

Rimuove tutti i caratteri all'interno di un insieme (così come ' ','\t','\r','\n') all'inizio di una stringa da questa stringa.

```
int  TrimLeft(
   const string  targets      // il set
   )
```

Parametri

targets

[in] Set di caratteri da rimuovere.

Valore di ritorno

Numero di caratteri rimossi.

TrimRight

Rimuove tutti i caratteri all'interno di un insieme (così come ' ','\t','\r','\n') alla fine di una stringa da questa stringa.

```
int  TrimRight(
   const string  targets      // il set
   )
```

Parametri

targets

[in] Set di caratteri da rimuovere.

Valore di ritorno

Numero di caratteri rimossi.

Clear

Cancella una stringa.

```
bool  Clear()
```

Valore di ritorno

true - in caso di successo, false - non può cancellare la stringa.

ToUpper

Converte tutti i caratteri della stringa in maiuscolo.

```
bool  ToUpper()
```

Valore di ritorno

true - in caso di successo, false - non si può convertire in maiuscolo.

ToLower

Converte tutti i caratteri della stringa in caratteri minuscoli.

```
bool  ToLower()
```

Valore di ritorno

true - in caso di successo, false - non si può convertire in minuscolo.

Reverse

Inverte una stringa (i caratteri iniziali e finali si scambiano in senso-pari).

```
void  Reverse()
```

Find

Cerca la prima corrispondenza di una stringa da una posizione specificata.

```
int  Find(
   uint          start,          // posizione
   const string  substring       // sottostringa
   ) const;
```

Parametri

start

[in] Posizione iniziale per la ricerca della sottostringa.

substring

[in] Sottostringa di esempio per la ricerca.

Valore di ritorno

L'indice della prima corrispondenza della sottostringa (-1 - la sottotringa non viene trovata).

FindRev

Cerca l'ultima corrispondenza della stringa.

```
int  FindRev(
   const string  substring      // sottostringa
   ) const;
```

Parametri

substring

[in] Sottostringa di esempio per la ricerca.

Valore di ritorno

L'indice dell'ultima corrispondenza della sottostringa (-1 - la sottostringa non è stata trovata).

Remove

Rimuove tutte le corrispondenze sottostringa.

```
uint  Remove(
   const string  substring      // sottostringa
   )
```

Parametri

substring

[in] Sottostringa di esempio per la ricerca.

Valore di ritorno

Numero di rimozioni sottostringa.

Replace

Sostituisce tutte le corrispondenze sottostringa.

```
uint  Replace(
   const string  substring,     // sottostringa
   const string  newstring      // sottostringa
   )
```

Parametri

substring

[in] Sottostringa di esempio per la ricerca.

newstring

[in] Sottostringa di esempio con cui sostituire.

Valore di ritorno

Numero di sostituzioni sottostringa.

Oggetti Grafici

Questa sezione contiene i dettagli tecnici di lavoro con classi di oggetti grafici ed una descrizione delle componenti rilevanti della Libreria MQL5 standard.

L'uso delle classi degli oggetti grafici farà risparmiare tempo quando si creeranno programmi personalizzati (script, expert).

La Libreria Standard MQL5 (in termini di oggetti grafici) si trova nella directory di lavoro del terminale nella cartella Include\ChartObjects.

Classe/Gruppo	Descrizione
CChartObject	Classe di base di un oggetto grafico
Linee	Gruppo di Classi "Linee"
Canali	Gruppo di Classi "Canali"
Attrezzi di Gann	Gruppo di Classi "Gann"
Attrezzi di Fibonacci	Gruppo di Classi "Fibonacci"
Attrezzi di Elliott	Gruppo di Classi "Elliott"
Shapes	Gruppo di Classi "Forme"
Frecce	Gruppo di Classi "Frecce"
Controlli	Gruppo di Classi "Controlli"

CChartObject

CChartObject è una classe base per le classi di oggetti grafici grafico della libreria Standard MQL5.

Descrizione

La Classe CChartObject fornisce l'accesso semplificato alle funzioni API MQL5 per tutti i suoi discendenti.

Dichiarazione

```
class CChartObject : public CObject
```

Titolo

```
#include <ChartObjects\ChartObject.mqh>
```

Gerarchia di ereditarietà

CObject

CChartObject

Discendenti diretti

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

I Metodi della Classe per Gruppi

Attributi	
ChartId	Ottiene l'ID del chart a cui appartiene un oggetto grafico
Window	Ottiene il numero di una finestra chart in cui si trova un oggetto grafico
Name	Ottiene/Imposta il nome di un oggetto grafico
NumPoints	Ottiene il numero di punti di ancoraggio
Assign	
Attach	Associa un oggetto grafico chart
SetPoint	Imposta i parametri del punto di ancoraggio
Delete	
Delete	Elimina un oggetto grafico chart
Detach	Distacca un oggetto grafico chart
Shift	

Attributi	
ShiftObject	Lo slittamento relativo dell'oggetto
ShiftPoint	Lo slittamento relativo di un oggetto punto
Object properties	
Time	Ottiene/Imposta le coordinate temporali di un punto oggetto
Price	Ottiene/Imposta la coordinata prezzo di un punto oggetto
Color	Ottiene/imposta il colore dell'oggetto
Style	Ottiene/Imposta lo stile dell'oggetto linea
Larghezza	Ottiene/Imposta la larghezza dell' oggetto llinea
BackGround	Ottiene/Imposta la flag di disegnare un oggetto sullo sfondo
Selected	Ottiene/Imposta la flag "selezionato" di un oggetto grafico
Selectable	Ottiene/Imposta la flag oggetto selezionabile
Descrizione	Ottiene/imposta il testo dell'oggetto
Tooltip	Ottiene/imposta il tooltip dell'oggetto
Timeframes	Ottiene/Imposta la maschera dell flag di visibilità oggetto
Z_Order	Ottiene/Imposta la priorità oggetto grafico per la ricezione di un evento di clic del mouse sul chart
CreateTime	Ottiene l'orario della creazione dell'oggetto
Proprietà del livello dell' oggetto	
LevelsCount	Ottiene/Imposta il numero di livelli dell'oggetto
LevelColor	Ottiene/Imposta il colore della linea del livello
LevelStyle	Ottiene/Imposta lo stile della linea del livello
LevelWidth	Ottiene/Imposta la larghezza della linea del livello
LevelValue	Ottiene/imposta il livello
LevelDescription	Ottiene/Imposta il testo del livello
L'accesso alle funzioni API MQL5	
GetInteger	Ottiene il valore della proprietà dell'oggetto
SetInteger	Imposta il valore della proprietà dell'oggetto
GetDouble	Ottiene il valore della proprietà dell'oggetto
SetDouble	Imposta il valore della proprietà dell'oggetto

Attributi	
GetString	Ottiene il valore della proprietà dell'oggetto
SetString	Imposta il valore della proprietà dell'oggetto
Input/Output	
virtual Save	Metodo virtuale di scrittura su un file
virtual Load	Metodo virtuale di lettura da un file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

ChartId

Ottiene l'ID del chart a cui appartiene un oggetto grafico.

```
long  ChartId() const
```

Valore di ritorno

ID del chart in cui si trova l'oggetto grafico. Se non vi è alcun oggetto associato, restituisce -1.

Esempio:

```
//--- esempio per CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene identificatore chart idintifier di un oggetto chart
   long chart_id=object.ChartId();
  }
```

Window

Ottiene l'indice della finestra chart in cui si trova l'oggetto grafico.

```
int  Window() const
```

Valore di ritorno

Il numero della finestra chart in cui si trova l'oggetto grafico (0 - finestra principale). Se non vi è alcun oggetto associato, restituisce -1.

Esempio:

```
//--- esempio per CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene la finestra dell'oggetto chart
   int window=object.Window();
  }
```

Name (Metodo Get)

Ottiene il nome dell'oggetto grafico.

```
string  Name() const
```

Valore di ritorno

Nome dell'oggetto grafico allegato ad un'istanza della classe. Se non c'è oggetto allegato, restituisce NULL.

Name (Metodo Set)

Imposta il nome dell'oggetto grafico.

```
bool  Name(
   string  name      // nuovo nome
   )
```

Parametri

name

[in] Il nuovo nome dell'oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare il nome.

Esempio:

```
//--- esempio per CChartObject::Name
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene il nome dell'oggetto chart
   string object_name=object.Name();
   if(object_name!="MyChartObject")
     {
     //--- imposta il nome dell'oggetto chart
     object.Name("MyChartObject");
     }
  }
```

NumPoints

Ottiene il numero di punti di ancoraggio di un oggetto grafico.

```
int  NumPoints() const
```

Valore di ritorno

Numero di punti che collegano un oggetto grafico allegato all'istanza della classe. Se non c'è oggetto allegato, restituisce 0.

Esempio:

```
//--- esempio per CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene il conteggio punti dell'oggetto chart
   int points=object.NumPoints();
  }
```

Attach

Associa un oggetto grafico a un'istanza della classe.

```
bool  Attach(
   long     chart_id,      // ID del chart
   string   name,          // nome dell'oggetto
   int      window,        // finestra chart
   int      points         // numero di punti
   )
```

Parametri

chart_id

[out] Identificatore Chart.

name

[in] Nome dell'oggetto grafico.

window

[in] Numero finestra Chart (0 - finestra principale).

points

[in] Numero di punti di ancoraggio dell'oggetto grafico.

Valore di ritorno

true - successo, false - non posso legare l'oggetto.

Esempio:

```
//--- esempio per CChartObject::Attach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- allega oggetto chart
   if(!object.Attach(ChartID(),"MyObject",0,2))
     {
      printf("Errore allegamento oggetto");
      return;
     }
  }
```

SetPoint

Imposta nuove coordinate del punto di ancoraggio specificato dell'oggetto grafico.

```
bool  SetPoint(
   int       point,          // numero del punto
   datetime  new_time,       // coordinate tempo
   double    new_price       // coordinate prezzo
   )
```

Parametri

point

[in] Numero del punto di ancoraggio dell'oggetto grafico.

new_time

[a] Nuovo valore per le coordinate tempo del punto di ancoraggio specificato.

new_price

[in] Nuovo valore per le coordinate prezzo del punto di ancoraggio specificato.

Valore di ritorno

true - successo, false - non posso cambiare le coordinate del punto.

Esempio:

```
//--- esempio per CChartObject::SetPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- imposta il punto dell'oggetto chart
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

Delete

Rimuove dal chart un oggetto grafico allegato.

```
bool  Delete()
```

Valore di ritorno

true - successo, false - non posso rimuovere l'oggetto.

Esempio:

```
//--- esempio per CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- distacca oggetto chart
   if(!object.Delete())
     {
      printf("Errore eliminazione oggetto");
      return;
     }
  }
```

Detach

Distacca l'oggetto grafico.

```
void  Detach()
```

Valore di ritorno

Nessuno.

Esempio:

```
//--- esempio per CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- distacca oggetto chart
   object.Detach();
  }
```

ShiftObject

Slitta un oggetto grafico.

```
bool  ShiftObject(
   datetime  d_time,      // incremento delle coordinate tempo
   double    d_price      // incremento delle coordinate prezzo
   )
```

Parametri

d_time

[in] Incremento dlle coordinate tempo di tutti i punti di ancoraggio.

d_price

[in] Incremento dlle coordinate prezzo di tutti i punti di ancoraggio.

Valore di ritorno

true - successo, false - non posso slittare l'oggetto.

Esempio:

```
//--- esempio per CChartObject::ShiftObject
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- slitta l'oggetto chart
   object.ShiftObject(d_time,d_price);
  }
```

ShiftPoint

Slitta un punto di ancoraggio specifico dell'oggetto grafico.

```
bool  ShiftPoint(
   int       point,       // numero punto
   datetime  d_time,      // incremento delle coordinate tempo
   double    d_price      // incremento delle coordinate prezzo
   )
```

Parametri

point

[in] Numero del punto di ancoraggio dell'oggetto grafico.

d_time

[in] Incremento delle coordinate tempo del punto specificato.

d_price

[in] Incremento delle coordinate prezzo del punto specificato.

Valore di ritorno

true - successo, false - non posso slittare il punto.

Esempio:

```
//--- esempio per CChartObject::ShiftPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- punto di slittamento per l'oggetto chart
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

Time (Metodo Get)

Ottiene la coordinata temporale del punto di ancoraggio specificato di un oggetto grafico.

```
datetime  Time(
   int  point      // numero punti
   ) const
```

Parametri

point

[in] Numero del punto di ancoraggio dell'oggetto grafico.

Valore di ritorno

Coordinata tempo del punto di ancoraggio specificato dell'oggetto grafico allegato ad un'istanza della classe. Se non vi è alcun oggetto allegato o l'oggetto non ha questo punto, restituisce 0.

Time (Metodo Set)

Imposta la coordinata temporale del punto di ancoraggio specificato di un oggetto grafico.

```
bool  Time(
   int       point,        // numero del punto
   datetime  new_time      // orario
   )
```

Parametri

point

[in] Numero del punto di ancoraggio dell'oggetto grafico.

new_time

[in] nuovo valore per la coordinata tempo del punto di ancoraggio dell' oggetto grafico specificato.

Valore di ritorno

true - successo, false - non si può cambiare la coordinata tempo.

Esempio:

```
//--- esempio per CChartObject::Time
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- ottiene l'orario del punto dell'oggetto chart
      datetime point_time=object.Time(i);
      if(point_time==0)
```

```
         {
          //--- imposta l'orario del punto dell'oggetto chart
          object.Time(i,TimeCurrent());
         }
      }
   }
```

Price (Metodo Get)

Ottiene le coordinate prezzo del punto di ancoraggio specificato di un oggetto grafico.

```
double  Price(
   int  point      // numero punti
   ) const
```

Parametri

point

[in] Numero del punto di ancoraggio dell'oggetto grafico.

Valore di ritorno

Coordinata prezzo del punto di ancoraggio specificato dell'oggetto grafico allegato all'istanza della classe. Se non vi è alcun oggetto allegato o l'oggetto non ha questo punto, restituisce EMPTY_VALUE.

Price (Metodo Set)

Imposta la coordinata prezzo del punto di ancoraggio specificato, di un oggetto grafico.

```
bool  Price(
   int     point,          // numero del punto
   double  new_price       // prezzo
   )
```

Parametri

point

[in] Numero del punto di ancoraggio dell'oggetto grafico.

new_price

[in] Nuovo valore per la coordinata prezzo del punto di ancoraggio dell' oggetto grafico specificato.

Valore di ritorno

true - successo, false - non si può cambiare la coordinata prezzo.

Esempio:

```
//--- esempio per CChartObject::Price
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- ottiene il prezzo del punto oggetto chart
      double point_price=object.Price(i);
```

```
      if(point_price!=price)
        {
         //--- imposta il prezzo del punto oggetto chart
         object.Price(i,price);
        }
     }
  }
```

Color (Metodo Get)

Ottiene il colore della linea dell'oggetto grafico.

```
color  Color() const
```

Valore di ritorno

Colore della linea dell'oggetto grafico allegato all'istanza della classe. Se non c'è oggetto allegato, restituisce CLR_NONE.

Color (Metodo Set)

Imposta il colore della linea dell'oggetto grafico.

```
bool  Color(
   color  new_color      // nuovo colore
   )
```

Parametri

new_color

[in] Nuovo valore del colore della linea dell'oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

Esempio:

```
//--- esempio per CChartObject::Color
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene il colore dell'oggetto chart
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- imposta il colore dell'oggetto chart
     object.Color(clrRed);
     }
  }
```

Style (Metodo Get)

Ottiene lo stile della linea dell'oggetto grafico.

```
ENUM_LINE_STYLE  Style() const
```

Valore di ritorno

Stile della linea dell' oggetto grafico allegato ad un'istanza della classe. Se non vi è alcun oggetto allegato, restituisce WRONG_VALUE.

Style (Metodo Set)

Imposta lo stile della linea dell'oggetto grafico.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // stile
   )
```

Parametri

new_style

[in] Nuovo valore dello stile dell'oggetto grafico linea.

Valore di ritorno

true - successo, false - non posso cambiare lo stile.

Esempio:

```
//--- esempio per CChartObject::Style
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene lo stile dell'oggetto chart
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- imposta lo stile dell'oggetto chart
      object.Style(STYLE_SOLID);
     }
  }
```

Width (Metodo Get)

Ottiene la larghezza della linea dell'oggetto grafico.

```
int  Width() const
```

Valore di ritorno

La larghezza della linea dell'oggetto grafico allegato ad un'istanza della classe. Se non vi è alcun oggetto allegato, restituisce -1.

Width (Metodo Set)

Imposta la larghezza della linea dell'oggetto grafico.

```
bool  Width(
   int  new_width      // spessore
   )
```

Parametri

new_width

[in] Nuovo valore dello spessore dell'oggetto grafico linea.

Valore di ritorno

true - successo, false - non posso cambiare la larghezza.

Esempio:

```
//--- esempio per CChartObject::Width
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene lo spessore dell'oggetto chart
   int width=object.Width();
   if(width!=1)
     {
      //--- imposta lo spessore dell'oggetto chart
      object.Width(1);
     }
  }
```

Background (Metodo Get)

Ottiene la flag per disegnare un oggetto grafico sullo sfondo.

```
bool  Background() const
```

Valore di ritorno

Flag per disegnare l'oggetto grafico, collegato ad un'istanza della classe, sullo sfondo. Se non c'è oggetto allegato, restituisce false.

Background (Metodo Set)

Imposta la flag per disegnare un oggetto grafico sullo sfondo.

```
bool  Background(
   bool  background      // valore del flag
   )
```

Parametri

background

[in] Nuovo valore del flag per disegnare un oggetto grafico sullo sfondo.

Valore di ritorno

true - successo, false - non posso cambiare il flag.

Esempio:

```
//--- esempio per CChartObject::Background
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene flag sfondo di un oggetto chart
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- imposta flag sfondo di un oggetto chart
     object.Background(true);
     }
  }
```

Selezionato (Metodo Get)

Ottiene il flag che indica che è stato selezionato un oggetto grafico. In altre parole - se l'oggetto grafico è selezionato o meno.

```
bool  Selected() const
```

Valore di ritorno

Per constatare che l'oggetto, collegato ad un'istanza della classe, è stato selezionato. Se non c'è oggetto allegato, restituisce false.

Selected (Metodo Set)

Imposta il flag che indica che l'oggetto grafico è selezionato.

```
bool  Selected(
   bool  selected      // valore del flag
   )
```

Parametri

selected

[in] Nuovo valore del flag che indica che è stato selezionato un oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare il flag.

Esempio:

```
//--- esempio per CChartObject::Selected
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene il flag "selected" dell'oggetto chart
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- imposta il flag "selected" dell'oggetto chart
     object.Selected(false);
     }
  }
```

Selectable (Metodo Get)

Ottiene la flag che indica la capacità di un oggetto grafico di essere selezionato. In altre parole - se l'oggetto grafico può essere selezionato o meno.

```
bool  Selectable() const
```

Valore di ritorno

Flag che indica la capacità di un oggetto grafico, collegato ad un'istanza della classe, di essere selezionato. Se non c'è oggetto allegato, restituisce false.

Selectable (Metodo Set)

Imposta la flag che indica la capacità di un oggetto grafico di essere selezionato.

```
bool  Selectable(
   bool  selectable      // valore del flag
   )
```

Parametri

selectable

[in] Nuovo valore del flag indica la capacità di un oggetto grafico di essere selezionato.

Valore di ritorno

true - successo, false - non posso cambiare il flag.

Esempio:

```
//--- esempio per CChartObject::Selectable
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene il flag "selezionabile" dell'oggetto chart
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- imposta il flag "selezionabile" dell'oggetto chart
     object.Selectable(false);
     }
  }
```

Description (Metodo Get)

Ottiene una descrizione (testo) di un oggetto grafico.

```
string  Description() const
```

Valore di ritorno

Descrizione (testo) dell'oggetto grafico allegato all'istanza della classe. Se non c'è oggetto allegato, restituisce NULL.

Description (Metodo Set)

Imposta la descrizione (testo) dell'oggetto grafico.

```
bool  Description(
   string  text      // testo
   )
```

Parametri

text

[in] Nuovo descrizione (testo) di un oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare la descrizione(testo).

Esempio:

```
//--- esempio per CChartObject::Description
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene la descrizione dell'oggetto chart
   string description=object.Description();
   if(description=="")
     {
      //--- imposta la descrizione dell'oggetto chart
      object.Description("MyObject");
     }
  }
```

Tooltip (Metodo Get)

Ottiene il tooltip testo di un oggetto grafico.

```
string  Tooltip() const
```

Valore di ritorno

Il tooltip testo di un oggetto grafico allegato ad un'istanza della classe. Se non c'è oggetto allegato, restituisce NULL.

Tooltip (Metoto Set)

Imposta il tooltip testo di un oggetto grafico.

```
bool  Tooltip(
   string  new_tooltip      // nuovo testo del tooltip
   )
```

Parametri

new_tooltip

[in] Nuovo testo di un tooltip di oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare il tooltip.

Note:

Se la proprietà non è impostata, allora viene visualizzato il tooltip generato automaticamente dal terminale. Un tootip può essere disabilitato impostando il valore "\n" (avanzamento riga).

Timeframes (Get Method)

Ottiene flafg di visibilità di un oggetto grafico.

```
int  Timeframes() const
```

Valore di ritorno

Flags di visibilità dell'oggetto grafico allegato ad un'istanza della classe. Se non c'è oggetto allegato, restituisce 0.

Timeframes (Metodo Set)

Imposta flag di visibilità di un oggetto grafico.

```
bool  Timeframes(
   int  new_timeframes      // flags di visibilità
   )
```

Parametri

new_timeframes

[in] Nuove flag di visibilità dell'oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare la visibilità della flag.

Esempio:

```
//--- esempio per CChartObject::Timeframes
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene il timeframe dell'oggetto chart
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //--- imposta il timeframe dell'oggetto chart
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

Z_Order (Get Method)

Ottiene la priorità oggetto grafico per la ricezione di un evento di clic del mouse sul chart (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

Valore di ritorno

La priorità di un oggetto grafico, allegato all' istanza della classe. Se non vi è alcun oggetto allegato, restituisce 0.

Z_Order (Metodo Set)

Imposta la priorità dell'oggetto grafico per la ricezione di un evento di clic del mouse sul chart (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // priorità oggetto grafico
   )
```

Parametri

value

[in] Nuovo valore della priorità di un oggetto grafico per ricevere l'evento CHARTEVENT_CLICK.

Valore di ritorno

true - successo, false - non si può cambiare la priorità.

Nota

La proprietà Z_Order gestisce una priorità durante la manipolazione di clic sugli oggetti grafici. Impostando il valore maggiore di 0 (valore di default), è possibile aumentare la priorità dell'oggetto durante la manipolazione di clic del mouse.

CreateTime

Ottiene l'orario di creazione dell'oggetto grafico.

```
datetime  CreateTime() const
```

Valore di ritorno

Ora di creazione dell'oggetto grafico allegato alla istanza della classe. Se non c'è oggetto allegato, restituisce 0.

Esempio:

```
//--- esempio per CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene l'orario di creazione per l'oggetto chart
   datetime create_time=object.CreateTime();
  }
```

LevelsCount (Metodo Get)

Ottiene il numero di livelli di un oggetto grafico.

```
int  LevelsCount() const
```

Valore di ritorno

Numero di livelli di oggetto grafico allegato ad un'istanza della classe. Se non c'è oggetto allegato, restituisce 0.

LevelsCount (Medoto Set)

Imposta il numero di livelli dell' oggetto grafico.

```
bool  LevelsCount(
   int  levels      // numero dei livelli
   )
```

Parametri

levels

[In] Il nuovo numero di livelli dell' oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare il numero di livelli.

Esempio:

```
//--- esempio per CChartObject::LevelsCount
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene conteggio livelli di un oggetto chart
   int levels_count=object.LevelsCount();
   //--- imposta conteggi livelli di un oggetto chart
   object.LevelsCount(levels_count+1);
  }
```

LevelColor (Metodo Get)

Ottiene il colore della linea di livello di un oggetto grafico specificato.

```
color  LevelColor(
   int  level      // numero livello
   ) const
```

Parametri

level

[in] Numero del livello di un oggetto grafico.

Valore di ritorno

Colore della linea del livello dell'oggetto grafico allegato alla istanza della classe specificata. Se non vi è alcun oggetto allegato o l'oggetto non ha il livello specificato, restituisce CLR_NONE.

LevelColor (Set Method)

Imposta il colore della linea del livello dell'oggetto grafico specificato.

```
bool  LevelColor(
   int    level,          // numero del livello
   color  new_color       // nuovo colore
   )
```

Parametri

level

[in] Numero del livello di un oggetto grafico.

new_color

[in] Nuovo colore della linea del livello di un oggetto grafico specificato.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

Esempio:

```
//--- esempio per CChartObject::LevelColor
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- ottiene il colore del livello dell'oggetto chart
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

```
         {
          //--- imposta il livello di colore dell'oggetto chart
          object.LevelColor(i,clrRed);
         }
      }
   }
```

LevelStyle (Metodo Get)

Ottiene lo stile della linea del livello di un oggetto grafico specificato.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // numero livello
   ) const
```

Parametri

level

[in] Numero del livello di un oggetto grafico.

Valore di ritorno

Stile linea del livello specificato dell' oggetto grafico allegato ad un'istanza della classe. Se non vi è alcun oggetto allegato o l'oggetto non ha il livello specificato, restituisce WRONG_VALUE.

LevelStyle (Metodo Set)

Imposta lo stile della linea del livello specificato dell'oggetto grafico.

```
int  LevelStyle(
   int              level,     // numero del livello
   ENUM_LINE_STYLE  style      // stile della linea
   )
```

Parametri

level

[in] Numero del livello di un oggetto grafico.

style

[in] Nuovo stile della linea del livello specificato di un oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare lo stile.

Esempio:

```
//--- esempio per CChartObject::LevelStyle
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- ottiene lo stile livelli dell'oggetto chart
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

```
         {
          //--- imposta lo stile del livello dell'oggetto chart
          object.LevelStyle(i,STYLE_SOLID);
         }
      }
   }
```

LevelWidth (Metodo Get)

Ottiene la larghezza della linea del livello di un oggetto grafico specificato.

```
int  LevelWidth(
   int  level      // numero livello
   ) const
```

Parametri

level

[in] Numero del livello di un oggetto grafico.

Valore di ritorno

Lo spessore della linea del livello del oggetto grafico allegato all' istanza della classe specificata. Se non vi è alcun oggetto collegato o l'oggetto non ha il livello specificato, restituisce -1.

LevelWidth (Metodo Set)

Imposta la larghezza della linea del livello dell'oggetto grafico specificato.

```
bool  LevelWidth(
   int  level,          // numero del livello
   int  new_width       // nuova larghezza
   )
```

Parametri

level

[in] Numero del livello di un oggetto grafico.

new_width

[in] Nuova larghezza della linea del livello specificato di un oggetto grafico.

Valore di ritorno

true - successo, false - non posso cambiare la larghezza.

Esempio:

```
//--- esempio per CChartObject::LevelWidth
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- ottiene lo spessore del livello dell'oggetto chart
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

```
         {
          //--- imposta lo spessore del livello dell'oggetto chart
          object.LevelWidth(i,1);
         }
      }
   }
```

LevelValue (Metodo Get)

Ottiene il valore del livello di un oggetto grafico specificato.

```
double  LevelValue(
   int  level      // numero livello
   ) const
```

Parametri

level

[in] Numero del livello di oggetto grafico.

Valore di ritorno

Il valore del livello di un oggetto grafico associato all'istanza della classe specificata. Se non vi è alcun oggetto associato o l'oggetto non ha un determinato livello, restituisce EMPTY_VALUE.

LevelValue (Metodo Set)

Imposta il valore del livello dell' oggetto grafico specificato.

```
bool  LevelValue(
   int     level,          // numero del livello
   double  new_value       // nuovo valore
   )
```

Parametri

level

[in] Numero del livello di oggetto grafico.

new_value

[In] Nuovo valore del livello di un oggetto grafico specificato.

Valore di ritorno

true - successo, false - non posso cambiare il valore.

Esempio:

```
//--- esempio per CChartObject::LevelValue
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- ottiene il valore del livello dell'oggetto chart
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

```
         {
          //--- imposta il valore del livello dell oggetto chart
          object.LevelValue(i,0.1*i);
         }
      }
   }
```

LevelDescription (Metodo Get)

Ottiene la descrizione (testo) del livello di un oggetto grafico.

```
string  LevelDescription(
   int  level       // numero del livello
   ) const
```

Parametri

level

[in] Numero del livello di oggetto grafico.

Valore di ritorno

Descrizione (testo) del livello di un oggetto grafico che è destinato ad un'istanza della classe specificata. Restituisce NULL se non vi è alcun oggetto associato o l'oggetto non ha un livello specificato.

LevelDescription (Metodo Set)

Imposta la descrizione (testo) del livello dell'oggetto grafico specificato.

```
bool  LevelDescription(
   int     level ,     // numero del livello
   string  text        // testo
   )
```

Parametri

level

[in] Numero del livello di oggetto grafico.

text

[in] Nuovo valore di descrizione (testo) del livello dell' oggetto grafico specificato.

Valore di ritorno

true - successo, false - non posso cambiare la descrizione (testo).

Esempio:

```
//--- esempio per CChartObject::LevelDescription
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- ottiene la descrizione del livello dell'oggetto chart
      string level_description=object.LevelDescription(i);
```

```
      if(level_description=="")
        {
         //--- ottiene la descrizione del livello dell'oggetto chart
         object.LevelDescription(i,"Level_"+IntegerToString(i));
        }
     }
  }
```

GetInteger

Fornisce l'accesso semplificato alle funzioni di API MQL5 ObjectGetInteger() per ricevere i valori delle proprietà integer (di tipo bool, char, char, short, ushort, int, uint, long, ulong, datetime, color) di un oggetto grafico legato ad un'istanza della classe. Ci sono due versioni della chiamata di funzione:

Ottenere un valore della proprietà senza controllare la correttezza

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // ID della proprietà integer
   int                           modifier=-1       // modificatore
   ) const
```

Parametri

prop_id

[in] ID dell'oggetto grafico proprietà double.

modifier=-1

[in] Modificatore (indice) di una proprietà double.

Valore di ritorno

Valore della proprietà integer - successo; 0 - se invece non può ricevere proprietà integer.

Ottenere il valore della proprietà verificando la correttezza del funzionamento

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // ID della proprietà integer
   int                           modifier,      // modificatore
   long&                         value          // link alla variabile
   ) const
```

Parametri

prop_id

[in] ID dell'oggetto grafico della proprietà integer.

modifier

[in] Modificatore (indice) della proprietà integer.

value

[out] Link alla variabile per piazzare in valore della proprietà integer.

Valore di ritorno

true - successo, false - non posso ottenere la proprietà integer.

Esempio:

```
//--- esempio per CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

```
{
 CChartObject object;
 //--- ottiene il colore del oggetto chart per metodo semplice
 printf("Il colore dell'oggetto è %s",ColorToString(object.GetInteger(OBJPROP_COLOR)
 //--- ottiene il colore dell'oggetto per metodo classico
 long color_value;
 if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
   {
    printf("Ottiene l'errore della proprietà integer %d",GetLastError());
    return;
   }
 else
    printf("Il colore dell'oggetto %s",color_value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- ottiene i livelli di spessore per metodo semplice
    printf("Livello %d spessore= %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
    //--- ottiene i livelli con il metodo classico
    long width_value;
    if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
      {
       printf("Ottiene l'errore della proprietà integer %d",GetLastError());
       return;
      }
    else
       printf("Livello %d spessore= %d",i,width_value);
   }
}
```

SetInteger

Fornisce l'accesso semplificato alle funzioni di API MQL5 ObjectSetInteger() per modificare le proprietà integer (di tipo bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) di un oggetto grafico destinato ad un' istanza della classe. Ci sono due versioni della chiamata di funzione:

Impostazione di un valore della proprietà che non richiede un modificatore

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // ID della proprietà integer
   long                           value         // valore
   )
```

Parametri

prop_id

[in] ID dell'oggetto grafico della proprietà integer.

value

[in] Nuovo valore di una proprietà integer modificata.

L'impostazione di un valore della proprietà che indica il modificatore

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // ID della proprietà integer
   int                            modifier,     // modifica
   long                           value         // valore
   )
```

Parametri

prop_id

[in] ID dell'oggetto grafico della proprietà integer.

modifier

[in] Modificatore (indice) della proprietà integer.

value

[in] Nuovo valore della proprietà integer.

Valore di ritorno

true - successo, false - non può cambiare la proprietà.

Esempio:

```
//--- esempio per CChartObject::SetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- imposta nuovi colori dell'oggetto chart
```

```
   if(!object.SetInteger(OBJPROP_COLOR,clrRed))
     {
      printf("Errore impostazione proprietà integer = %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- imposta spessore livelli
      if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
        {
         printf("Errore imposta proprietà integer = %d",GetLastError());
         return;
        }
     }
  }
```

GetDouble

Fornisce l'accesso semplificato alle funzioni di API MQL5 ObjectGetDouble() per ricevere valori double (di tipo float e double) di un oggetto grafico legato ad un'istanza della classe. Ci sono due versioni della chiamata di funzione:

Ottenere un valore della proprietà senza controllare la correttezza

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // ID della proprietà integer
   int                          modifier=-1      // modificatore
   ) const
```

Parametri

prop_id

[in] ID dell'oggetto grafico proprietà double.

modifier=-1

[in] Modificatore (indice) di una proprietà double.

Valore di ritorno

Valore di una proprietà double - successo; EMPTY_VALUE - non può ricevere la proprietà double.

Ottenere il valore della proprietà in verifica della correttezza di tale trattamento

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,       // ID della proprietà double
   int                          modifier,      // modificatore
   double&                      value          // link alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà double di un oggetto grafico.

modifier

[in] Modificatore (indice) di una proprietà double.

value

[out] Link alla variabile in cui piazzare il valore della proprietà double.

Valore di ritorno

true - successo, false - non posso ottenere la proprietà double.

Esempio:

```
//--- esempio per CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

```
{
 CChartObject object;
 //---
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- ottiene i valori dei livelli per metodo semplice
    printf("Livello %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
    //--- ottiene i valori dei livelli dal metodo classico
    double value;
    if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
      {
       printf("Ottiene l'errore della proprietà double %d",GetLastError());
       return;
      }
    else
       printf("Livello %d valore=%f",i,value);
   }
}
```

SetDouble

Fornisce l'accesso semplificato alle funzioni di API MQL5 ObjectSetDouble() per modificare le proprietà double (di tipi float e double) di un oggetto grafico legato ad un'istanza della classe. Ci sono due versioni della chiamata di funzione:

Impostazione di un valore della proprietà che non richiede un modificatore

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // ID della proprietà double
   double                        value         // valore
   )
```

Parametri

prop_id

[in] ID della proprietà double di un oggetto grafico.

value

[in] Nuovo valore delle proprietà double cambiata.

L'impostazione di un valore della proprietà che indica il modificatore

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,       // ID della proprietà double
   int                           modifier,      // modificatore
   double                        value          // valore
   )
```

Parametri

prop_id

[in] ID della proprietà double di un oggetto grafico.

modifier

[in] Modificatore (indice) di una proprietà double.

value

[in] Nuovo valore delle proprietà double cambiata.

Valore di ritorno

true - successo, false - non può cambiare la caratteristica double.

Esempio:

```
//--- esempio per CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
```

```
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- imposta i valori dei livelli dell'oggetto chart
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
        {
         printf("Errore imposta proprietà double = %d",GetLastError());
         return;
        }
     }
  }
```

GetString

Fornisce l'accesso semplificato alle funzioni di API MQL5 ObjectGetString() per i valori della proprietà string di un oggetto grafico destinati ad un'istanza della classe. Ci sono due versioni della chiamata di funzione:

Ottenere un valore della proprietà senza controllare la correttezza

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // ID della proprietà stringa
   int                          modifier=-1      // modificatore
   ) const
```

Parametri

prop_id

[in] ID della proprietà string di un oggetto grafico.

modifier=-1

[in] Modificatore (indice) di una proprietà string.

Valore di ritorno

Valore della proprietà string - successo "" - non può ricevere la proprietà string.

Ottenere un valore della proprietà verificando la correttezza di tale trattamento

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // ID della proprietà string
   int                          modifier,      // modificatore
   string&                      value          // link alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà string di un oggetto grafico.

modifier

[in] Modificatore (indice) di una proprietà string.

value

[out] Link alla variabile in cui piazzare il valore della proprietà string.

Valore di ritorno

true - successo, false - non posso ottenere la proprietà string.

Esempio:

```
//--- esempio per CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

```
{
 CChartObject object;
 string       value;
 //--- ottiene il nome dell'ogeetto chart per metodo semplice
 printf("Nome dell'oggetto='%s'",object.GetString(OBJPROP_NAME));
 //--- ottiene il nome dell'oggetto chart per metodo classico
 if(!object.GetString(OBJPROP_NAME,0,value))
   {
    printf("Ottiene l'errore proprietà stringa %d",GetLastError());
    return;
   }
 else
    printf("Il nome dell'oggetto è '%s'",value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- ottiene descrizione livelli per metodo semplice
    printf("Livello %d descrizione = '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
    //--- ottiene descrizione livelli per metodo classico
    if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
      {
       printf("Ottiene errore proprietà string %d",GetLastError());
       return;
      }
    else
       printf("Livello %d descrizione = '%s'",i,value);
   }
}
```

SetString

Fornisce l'accesso semplificato alle funzioni di API MQL5 ObjectSetString() per cambiare le proprietà string di un oggetto grafico legato ad un'istanza della classe. Ci sono due versioni della chiamata di funzione:

Impostazione di un valore della proprietà che non richiede un modificatore

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // ID proprietà string
   string                        value         // valore
   )
```

Parametri

prop_id

[in] ID della proprietà string di un oggetto grafico.

value

[In] Nuovo valore della proprietà di stringa modificata.

L'impostazione di un valore della proprietà che indica il modificatore

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,       // ID della proprietà string
   int                           modifier,      // modificatore
   string                        value          // valore
   )
```

Parametri

prop_id

[in] ID della proprietà string di un oggetto grafico.

modifier

[in] Modificatore (indice) di una proprietà string.

value

[In] Nuovo valore della proprietà di stringa modificata.

Valore di ritorno

true - successo, false - non posso cambiare la proprietà string.

Esempio:

```
//--- esempio per CChartObject::SetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- imposta il nuovo nome dell'oggetto chart
```

```
   if(!object.SetString(OBJPROP_NAME,"MyObject"))
     {
      printf("Imposta l'errore della proprietà string %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- imposta descrizione dei livelli
      if(!object.SetString(OBJPROP_LEVELTEXT,i,"Level_"+IntegerToString(i)))
        {
         printf("Imposta l'errore della proprietà string %d",GetLastError());
         return;
        }
     }
  }
```

Save

Salva parametri dell'oggetto nel file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file precedentemente aperto con la funzione FileOpen (...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CChartObject::Save
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object=new CChartObject;
   //--- imposta parametri oggetto
   //--- . . .
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva File: Errore %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

Load

Carica i parametri dell'oggetto dal file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - completato con successo, false - errore.

Esempio:

```
//--- esempio per CChartObject::Load
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object;
   //--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- errore caricamento file
         printf("Caricamento file: Errore %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- usa oggetto
   //--- . . .
  }
```

Type

Ottiene l' ID del tipo di oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Object type ID (0x8888 for CChartObject).

Esempio:

```
//--- esempio per CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- ottiene il tipo dell'oggetto
   int type=object.Type();
  }
```

Oggetti Linea

Gruppo di oggetti grafici "Lines" (Linee).

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Lines" (Linee) e una descrizione dei componenti rilevanti della libreria standard MQL5.

Nome della classe	Oggetto
CChartObjectVLine	"Vertical Line" (oggetto grafico "Linea Verticale")
CChartObjectHLine	"Horizontal Line" (oggetto grafico "Linea Orizzontale")
CChartObjectTrend	"Trend Line" (oggetto grafico "Linea di Tendenza(Trend)")
CChartObjectTrendByAngle	"Trend Line by Angle" (oggetto grafico "Linea di Tendenza per Angolo")
CChartObjectCycles	"Cyclic Lines" (oggetto grafico "Linee Cicliche")

Vedi anche

Tipi di oggetti, Oggetti grafici

CChartObjectVLine

CChartObjectVLine è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Vertical Line"("linea verticale").

Descrizione

La Classe CChartObjectVLine fornisce l'accesso alle proprietà dell'oggetto "Vertical Line".

Dichiarazione

```
class CChartObjectVLine : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectVLine

I Metodi della Classe per Gruppi

Create	
Create	Crea l' oggetto grafico "Vertical Line"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea oggetto grafico "Vertical Line".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time           // coordinate tempo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time

[in] Coordinate orarie del punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Tipo identificatore oggetto (OBJ_VLINE for CChartObjectVLine).

CChartObjectHLine

CChartObjectHLine è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Horizontal Line" ("Linea Orizzontale").

Descrizione

La Classe CChartObjectHLine fornisce l'accesso alle proprietà dell'oggetto "Horizonal Line".

Dichiarazione

```
class CChartObjectHLine : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectHLine

I Metodi della Classe per Gruppi

Create	
Create	Crea oggetto grafico "Linea Orizzontale"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Linea orizzontale".

```
bool  Create(
   long    chart_id,      // identificatore chart
   string  name,          // nome oggetto
   long    window,        // finestra del chart
   double  price          // coordinate prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

price

[in] Coordinate prezzo del punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Tipo identificatore oggetto (OBJ_HLINE per CChartObjectHLine).

CChartObjectTrend

CChartObjectTrend è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Trend Line".

Descrizione

La Classe CChartObjectTrend fornisce l'accesso alle proprietà dell'oggetto "Trend Line".

Dichiarazione

```
class CChartObjectTrend : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend

Discendenti diretti

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Trend Line"
Proprietà	
RayLeft	Ottiene/Imposta la proprietà "Ray Left"
RayRight	Ottiene/Imposta la proprietà "Ray Right"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Trend Line".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // Coordinate 1mo orario
   double    price1,        // Coordinate primo prezzo
   datetime  time2,         // Coordinate 2ndo orario
   double    price2         // Coordinate 2ndo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

RayLeft (Metodo Get)

Ottiene il valore della "proprietà Raggio a Sinistra".

```
bool  RayLeft() const
```

Valore di ritorno

Il valore della proprietà "Ray Left" assegnata all'istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

RayLeft (Metodo Set)

Imposta nuovo valore flag per la proprietà "Ray Left".

```
bool  RayLeft(
   bool  ray      // flag
   )
```

Parametri

ray

[in] Nuovo valore della proprietà "Ray Left".

Valore di ritorno

true - successo, false - non si può cambiare la flag.

RayRight (Metodo Get)

Ottiene il valore della proprietà "Ray Right".

```
bool  RayRight() const
```

Valore di ritorno

Il valore della proprietà "Ray Right", assegnato all'istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

RayRight (Metodo Set)

Imposta nuovo valore bandiera per la proprietà "Ray Right".

```
bool  RayRight(
   bool  ray      // flag
   )
```

Parametri

ray

[in] Nuovo valore della proprietà "Ray Right".

Valore di ritorno

true - successo, false - non si può cambiare la flag.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle di un file aperto in precedenza utilizzando la funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file aperto in precedenza utilizzando la funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Tipo di identificatore oggetto (OBJ_TREND per CChartObjectTrend).

CChartObjectTrendByAngle

CChartObjectTrendByAngle è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Trend Line by Angle" (Trend Line per Angolo).

Descrizione

La Classe CChartObjectTrendByAngle fornisce l'accesso alle proprietà dell'oggetto "Trend Line by Angle" .

Dichiarazione

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle

Discendenti diretti

CChartObjectGannLine

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Trend Line by Angle"
Proprietà	
Angle	Ottiene/Imposta la proprietà "Angolo"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Trend Line by Angle".

```
bool  Create(
   long      chart_id,     // identificatore chart
   string    name,         // nome dell'oggetto
   long      window,       // finestra chart
   datetime  time1,        // Coordinate 1mo orario
   double    price1,       // Coordinate primo prezzo
   datetime  time2,        // Coordinate 2ndo orario
   double    price2        // Coordinate 2ndo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Angle (metodo Get)

Ottiene il valore della proprietà "Angle"(angolo).

```
double  Angle() const
```

Valore di ritorno

Il valore della proprietà "Angle" assegnata all' istanza della classe. Se non c'è oggetto assegnato, restituisce Empty_value.

Angle (Metodo Set)

Imposta nuovo valore per la proprietà il "Angle".

```
bool  Angle(
   double  angle      // angolo
   )
```

Parametri

angle

[in] Nuovo valore della proprietà "Angle".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_TRENDBYANGLE for CChartObjectTrendByAngle).

CChartObjectCycles

CChartObjectCycles è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Cyclic Lines".

Descrizione

La CChartObjectCycles fornisce l'accesso alle proprietà dell'oggetto "Cyclic Lines".

Dichiarazione

```
class CChartObjectCycles : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectCycles

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Cycle Lines"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Cyclic Lines".

```
bool  Create(
   long      chart_id,     // identificatore chart
   string    name,         // nome dell'oggetto
   long      window,       // finestra chart
   datetime  time1,        // Coordinate 1mo orario
   double    price1,       // Coordinate primo prezzo
   datetime  time2,        // Coordinate 2ndo orario
   double    price2        // Coordinate 2ndo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_CYCLES per CChartObjectCycles).

Oggetti Canale

Un gruppo di oggetti grafici "Channels" (canali).

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Channels" (canali) e una descrizione dei componenti rilevanti della libreria standard MQL5.

Nome della classe	Oggetto
CChartObjectChannel	Oggetto grafico "Equidistant Channel" (Canale Equidistante)
CChartObjectRegression	Oggetto Grafico "Linear Regression Channel" (Canale Regressione Lineare)
CChartObjectStdDevChannel	Oggetto Grafico "Standard deviations Channel" (Canale di Deviazione Standard)
CChartObjectPitchfork	Oggetto Grafico "Andrew's Pitchfork" (Forche di Andrews)

Vedi anche

Tipi di oggetti, Oggetti grafici

CChartObjectChannel

CChartObjectChannel è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Canale Equidistante".

Descrizione

La Classe CChartObjectChannel fornisce l'accesso alle proprietà dell'oggetto "Canale Equidistante".

Dichiarazione

```
class CChartObjectChannel : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectChannel

I Metodi della Classe per Gruppi

Create	
Create	Crea oggetto grafico "Canale Equidistante"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea oggetto grafico "Canale Equidistante" .

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate tempo del primo punto di ancoraggio
   double    price1,        // coordinate prezzo del primo punto di ancoraggio
   datetime  time2,         // coordinate tempo del secondo punto di ancoraggio
   double    price2,        // coordinate prezzo del secondo punto di ancoraggio
   datetime  time3,         // coordinate tempo del terzo punto di ancoraggio
   double    price3         // coordinate prezzo del terzo punto di ancoraggio
   )
```

Parametri

chart_id

[in] Chart ID (0 - corrente chart).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Prezzo per il terzo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Object type identifier (OBJ_CHANNEL for CChartObjectChannel).

CChartObjectRegression

La Classe CChartObjectRegression è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Linear Regression Channel".

Descrizione

La Classe CChartObjectRegression fornisce l'accesso alle proprietà oggetto "Linear Regression Channel".

Dichiarazione

```
class CChartObjectRegression : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend
CChartObjectRegression

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Linear Regression Channel"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Linear Regression Channel".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   long      window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   datetime  time2          // seconde coordinate orario
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_REGRESSION for CChartObjectRegression).

CChartObjectStdDevChannel

CChartObjectStdDevChannel è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Standard Deviation Channel" (Canale di Deviazione Standard).

Descrizione

Classe CChartObjectStdDevChannel fornisce l'accesso alle proprietà dell'oggetto "Standard Deviation Channel".

Dichiarazione

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectStdDevChannel

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Standard Deviation Channel"
Proprietà	
Deviations	Ottiene/Imposta la proprietà "Deviazione"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Standard Deviation Channel".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // prime coordinate orario
   datetime  time2,         // seconde coordinate orario
   double    deviation      // deviazione
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

deviation

[in] Valore numerico per le proprietà "Deviation".

Valore di ritorno

true - successo, false - errore.

Deviazioni (Metodo Get)

Ottiene il valore della proprietà "Deviation".

```
double  Deviation() const
```

Valore di ritorno

Valore della proprietà "Deviazione"(Deviation) assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce Empty_value.

Deviazioni (Metodo Set)

Imposta il valore della proprietà "Deviazione".

```
bool  Deviation(
   double  deviation      // deviazione
   )
```

Parametri

deviation

[in] Nuovo valore per la proprietà "Deviation".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file aperto in precedenza tramite la funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file aperto in precedenza tramite la funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_STDDEVCHANNEL per CChartObjectStdDevChannel).

CChartObjectPitchfork

CChartObjectPitchfork è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Andrew Pitchfork".

Descrizione

La Classe CChartObjectPitchfork fornisce l'accesso alle proprietà dell'oggetto grafico "Andrew Pitchfork".

Dichiarazione

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend
CChartObjectPitchfork

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Andrew's Pitchfork"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Andrew's Pitchfork" .

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   long      window,        // finestra chart
   datetime  time1,         // coordinate tempo del primo punto di ancoraggio
   double    price1,        // coordinate prezzo del primo punto di ancoraggio
   datetime  time2,         // coordinate tempo del secondo punto di ancoraggio
   double    price2,        // coordinate prezzo del secondo punto di ancoraggio
   datetime  time3,         // coordinate tempo del terzo punto di ancoraggio
   double    price3         // coordinate prezzo del terzo punto di ancoraggio
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Nome univoco dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Prezzo per il terzo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_PITCHFORK for CChartObjectPitchfork).

Gann Tools(Strumenti di Gann)

Gruppo di oggetti grafici "Strumenti di Gann".

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Strumenti di Gann" e una descrizione dei componenti rilevanti della libreria standard MQL5.

Nome della classe	Oggetto
CChartObjectGannLine	"Gann Line" (oggetto grafico "Linea di Gann")
CChartObjectGannFan	"Gann Fan" (oggetto grafico "Ventaglio di Gann")
CChartObjectGannGrid	"Gann Grid" (oggetto grafico "Griglia di Gann")

Vedi anche

Tipi di oggetti, Oggetti grafici

CChartObjectGannLine

La Classe CChartObjectGannLine è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Gann Line".

Descrizione

La Classe CChartObjectGannLine fornisce l'accesso alle proprietà dell'oggetto "Gann Line".

Dichiarazione

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Titolo

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle
CChartObjectGannLine

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Gann Line"
Proprietà	
PipsPerBar	Ottiene/Imposta la proprietà "Pips per barra"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Metodi ereditati dalla classe CChartObjectTrendByAngle

Angle, Angle, Create

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Create l'oggetto grafico "Gann Line".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    ppb            // pips per barra
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

ppb

[in] Pips per barra.

Valore di ritorno

true - successo, false - errore.

PipsPerBar (Metodo Get)

Ottiene il valore della proprietà "Pips per barra".

```
double  PipsPerBar() const
```

Valore di ritorno

Valore di "Pips per barra"proprietà dell'oggetto assegnato alla istanza della classe. Se non c'è un oggetto assegnato, restituisce EMPTY_VALUE.

PipsPerBar (Metodo Set)

Imposta nuovo valore per la proprietà "Pips per barra".

```
bool  PipsPerBar(
   double  ppb      // pips per barra
   )
```

Parametri

ppb

[In] Nuovo valore per la proprietà "Pips per barra".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto utilizzando la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto utilizzando la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_GANNLINE per CChartObjectGannLine).

CChartObjectGannFan

La Classe CChartObjectGannFan è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Gann Fan".

Descrizione

La Classe CChartObjectGannFan fornisce l'accesso alle proprietà dell'oggetto "Gann Fan".

Dichiarazione

```
class CChartObjectGannFan : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend
CChartObjectGannFan

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Gann Fan"
Proprietà	
PipsPerBar	Ottiene/Imposta la proprietà "Pips per barra"
Downtrend	Ottiene/Imposta la proprietà "Downtrend"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Cra l'oggetto grafico "Gann Fan" .

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    ppb            // pips per barra
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra di base).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

ppb

[in] Pips per barra.

Valore di ritorno

true - successo, false - errore.

PipsPerBar (Metodo Get)

Ottiene il valore della proprietà "pips per barra".

```
double  PipsPerBar() const
```

Valore di ritorno

Il valore della proprietà "pips per barra" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce Empty_value.

PipsPerBar (Metodo Set)

Imposta nuovo valore per la proprietà "Pips per barra".

```
bool  PipsPerBar(
   double  ppb      // pips per barra
   )
```

Parametri

ppb

[in] Nuovo valore per la proprietà "Pips per barra".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Downtrend (Metodo Get)

Ottiene il valore del flag "Downtrend".

```
bool  Downtrend() const
```

Valore di ritorno

Valore della flag "Downtrend" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

Downtrend (Metodo Set)

Imposta il nuovo valore della proprietà "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // valore flag
   )
```

Parametri

downtrend

[in] Nuovo valore per la proprietà "Downtrend".

Valore di ritorno

true - successo, false - non posso cambiare il flag.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_GANNFAN for CChartObjectGannFan).

CChartObjectGannGrid

CChartObjectGannGrid è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Gann Grid".

Descrizione

La Classe CChartObjectGannGrid fornisce l'accesso alle proprietà dell'oggetto "Gann Grid".

Dichiarazione

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannGrid

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Gann Grid"
Proprietà	
PipsPerBar	Ottiene/Imposta la proprietà "Pips per barra"
Downtrend	Ottiene/Imposta la proprietà "Downtrend"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Gann Grid".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    ppb            // pips per barra
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

ppb

[in] Pips per barra.

Valore di ritorno

true - successo, false - errore.

PipsPerBar (Metodo Get)

Ottiene il valore della proprietà "pips per barra".

```
double  PipsPerBar() const
```

Valore di ritorno

Il valore della proprietà "pips per barra" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce Empty_value.

PipsPerBar (Metodo Set)

Imposta nuovo valore per la proprietà "Pips per barra".

```
bool  PipsPerBar(
   double  ppb       // Pips per barra
   )
```

Parametri

ppb

[in] Nuovo valore per la proprietà "Pips per barra".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Downtrend (Metodo Get)

Ottiene il valore della proprietà "Downtrend" (tendenza al ribasso)".

```
bool  Downtrend() const
```

Valore di ritorno

Valore della proprietà dell'oggetto "Downtrend" assegnata alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

Downtrend (Metodo Set)

Imposta il nuovo valore della proprietà "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // valore flag
   )
```

Parametri

downtrend

[in] Nuovo valore per la proprietà "Downtrend".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Type

Restituisce l'identificatore di tipo di oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_GANNGRID per CChartObjectGannGrid).

Fibonacci Tools(Strumenti di Fibonacci)

Gruppo di oggetti grafici "Strumenti di Fibonacci".

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Strumenti di Fibonacci" e una descrizione dei componenti rilevanti della libreria standard MQL5.

Nome della classe	Oggetto
CChartObjectFibo	"Fibonacci Retracement" (oggetto grafico "Ritracciamenti di Fibonacci")
CChartObjectFiboTimes	"Fibonacci Time Zones" (oggetto grafico "Zone Temporali di Fibonacci")
CChartObjectFiboFan	"Fibonacci Fan" (oggetto grafico "Ventaglio di Fibonacci")
CChartObjectFiboArc	"Fibonacci Arc" (oggetto grafico "Arco di Fibonacci")
CChartObjectFiboChannel	"Fibonacci Channel" (oggetto grafico "Canale di Fibonacci")
CChartObjectFiboExpansion	"Fibonacci Expansion" (oggetto grafico "Espansione di Fibonacci")

Vedi anche

Tipi di oggetti, Oggetti grafici

CChartObjectFibo

La classe CChartObjectFibo è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Ritracciamento di Fibonacci".

Descrizione

Classe CChartObjectFibo fornisce l'accesso alle proprietà dell'oggetto grafico "Ritracciamento di Fibonacci".

Dichiarazione

```
class CChartObjectFibo : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend
CChartObjectFibo

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Ritracciamento di Fibonacci"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Ritracciamento di Fibonacci" .

```
bool  Create(
   long      chart_id,     // identificatore chart
   string    name,         // nome dell'oggetto
   int       window,       // finestra chart
   datetime  time1,        // coordinate del primo orario
   double    price1,       // coordinate del primo prezzo
   datetime  time2,        // coordinate del secondo orario
   double    price2        // coordinate del secondo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_FIBO for CChartObjectFibo).

CChartObjectFiboTimes

CChartObjectFiboTimes è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Fibonacci Time Zones".

Descrizione

La Classe CChartObjectFiboTimes fornisce l'accesso alle proprietà dell'oggetto "Fibonacci Time Zones".

Dichiarazione

```
class CChartObjectFiboTimes : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectFiboTimes

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Fibonacci Time Zones"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Fibonacci Time Zones".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2         // coordinate del secondo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_FIBOTIMES per CChartObjectFiboTimes).

CChartObjectFiboFan

CChartObjectFiboFan è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Fibonacci Fan".

Descrizione

La Classe CChartObjectFiboFan fornisce l'accesso alle proprietà dell'oggetto "Fibonacci Fan".

Dichiarazione

```
class CChartObjectFiboFan : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectFiboFan

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Fibonacci Fan"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Creates "Fibonacci Fan" graphical object.

```
bool  Create(
   long      chart_id,     // identificatore chart
   string    name,         // nome dell'oggetto
   int       window,       // finestra chart
   datetime  time1,        // coordinate del primo orario
   double    price1,       // coordinate del primo prezzo
   datetime  time2,        // coordinate del secondo orario
   double    price2        // coordinate del secondo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_FIBOFAN per CChartObjectFiboFan).

CChartObjectFiboArc

La classe CChartObjectFiboArc è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Fibonacci Arc".

Descrizione

La Classe CChartObjectFiboArc fornisce l'accesso alle proprietà dell'oggetto "Fibonacci Arc".

Dichiarazione

```
class CChartObjectFiboArc : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectFiboArc

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Fibonacci Arc"
Proprietà	
Scale	Ottiene/Imposta la proprietà "Scale"
Ellisse	Ottiene/Imposta la proprietà "Ellisse"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Fibonacci Arc"

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2,        // coordinate del secondo prezzo
   double    scale          // scale (scala)
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

scale

[in] Scale.

Valore di ritorno

true - successo, false - errore.

Scale (Metodo Get)

Ottiene il valore della proprietà "Scala".

```
double  Scale() const
```

Valore di ritorno

Valore della proprietà dell'oggetto "Scale" assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce Empty_value.

Scale (Metodo Set)

Imposta nuovo valore per la proprietà "Scale".

```
bool  Scale(
   double  scale      // scale
   )
```

Parametri

scale

[in] Nuovo valore per la proprietà "Scale".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Ellisse (metodo Get)

Ottiene il valore del flag "Ellisse".

```
bool  Ellipse() const
```

Valore di ritorno

Valore del flag "Ellipse" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

Ellisse (Metodo Set)

Imposta il valore del flag "Ellisse".

```
bool  Ellipse(
   bool  ellipse      // valore flag
   )
```

Parametri

ellipse

[in] Nuovo valore per la proprietà "Ellisse".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_FIBOARC for CChartObjectFiboArc).

CChartObjectFiboChannel

La classe CChartObjectFiboChannel è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Fibonacci Channel".

Descrizione

La Classe CChartObjectFiboChannel fornisce l'accesso alle proprietà dell'oggetto "Canale di Fibonacci".

Dichiarazione

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboChannel

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Fibonacci Channel"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico"Fibonacci Channel".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2,        // coordinate del secondo prezzo
   datetime  time3,         // coordinate del terzo orario
   double    price3         // coordinate del terzo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Prezzo per il terzo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_FIBOCHANNEL per CChartObjectFiboChannel).

CChartObjectFiboExpansion

CChartObjectFiboExpansion è una classe per l'accesso semplificato alle proprietà dell'oggetto "Fibonacci Expansion".

Descrizione

La Classe CChartObjectFiboExpansion fornisce l'accesso alle proprietà dell'oggetto "Fibonacci Expension".

Dichiarazione

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Titolo

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectTrend
CChartObjectFiboExpansion

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Fibonacci Expansion"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Fibonacci Expansion".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2,        // coordinate del secondo prezzo
   datetime  time3,         // coordinate del terzo orario
   double    price3         // coordinate del terzo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Prezzo per il terzo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_EXPANSION per CChartObjectFiboExpansion).

Elliott Tools(Strumenti di Elliott)

Gruppo di oggetti grafici "Strumenti di Elliott".

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Attrezzi di Elliot" (Strumenti di Elliot).

Nome della classe	Oggetto
CChartObjectElliottWave3	"Correcting Wave" (oggetto grafico "Onda Correttiva")
CChartObjectElliottWave5	"Impulse Wave" (oggetto grafico "Onda Impulsiva")

Vedi anche

Tipi di oggetti, Oggetti grafici

CChartObjectElliottWave3

CChartObjectElliottWave3 è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Correcting Wave" (Onda Correttiva).

Descrizione

La Classe CChartObjectElliottWave3 fornisce l'accesso alle proprietà dell'oggetto grafico "Correcting Wave".

Dichiarazione

```
class CChartObjectElliottWave3 : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Gerarchia di ereditarietà

CObject

CChartObject

CChartObjectElliottWave3

Discendenti diretti

CChartObjectElliottWave5

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Correcting Wave"
Proprietà	
Grado	Ottiene/Imposta la proprietà "Degree"(grado)
Linee	Ottiene/Imposta la proprietà "Linee"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Correcting Wave".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2,        // coordinate del secondo prezzo
   datetime  time3,         // coordinate del terzo orario
   double    price3         // coordinate del terzo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Degree (Metodo Get)

Ottiene il valore della proprietà "Degree" (grado).

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

Valore di ritorno

Valore della proprietà "Grado" dell'oggetto assegnato all' istanza della classe. Se non c'è alcun oggetto assegnato, restituisce WRONG_VALUE.

Degree (Metodo Set)

Imposta il nuovo valore per la proprietà "Degree".

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // valore della proprietà
   )
```

Parametri

degree

[in] Nuovo valore per la proprietà "Degree".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Lines (Metodo Get)

Ottiene il valore della proprietà di "Linee".

```
bool  Lines() const
```

Valore di ritorno

Valore della proprietà "linee" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

Lines (Metodo Set)

Imposta nuovo valore per la proprietà "Linee".

```
bool  Lines(
   bool  lines      // valore flag
   )
```

Parametri

lines

[in] Nuovo valore per la proprietà "Linee".

Valore di ritorno

true - successo, false - non posso cambiare il flag.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] Handle del file binario già aperto con la funzione FileOpen(...)

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'oggetto (OBJ_ELLIOTWAVE3 per CChartObjectElliottWave3).

CChartObjectElliottWave5

CChartObjectElliottWave5 è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Wave Impulse".

Descrizione

La Classe CChartObjectElliottWave5 fornisce l'accesso alle proprietà dell'oggetto "Impulse Wave".

Dichiarazione

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Titolo

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

I Metodi della Classe per Gruppi

Create	
Create	Crea l' oggetto grafico "Impulse Wave"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectElliottWave3

Degree, Degree, Lines, Lines, Create, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Impulse Wave".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2,        // coordinate del secondo prezzo
   datetime  time3,         // coordinate del terzo orario
   double    price3,        // coordinate del terzo prezzo
   datetime  time4,         // coordinate del quarto orario
   double    price4,        // coordinate del quarto prezzo
   datetime  time5          // coordinate del quinto orario
   double    price5,        // coordinate del quinto prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Prezzo per il terzo punto di ancoraggio.

time4

[in] Coordinate orarie per il quarto punto di ancoraggio.

price4

[in] Coordinate prezzo per il quarto punto di ancoraggio.

time5

[in] Coordinate orarie per il quinto punto di ancoraggio.

price5

[in] Coordinate prezzo per il quinto punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del Tipo d'Oggetto (OBJ_ELLIOTWAVE5 for CChartObjectElliottWave5).

Shape Objects

Gruppo di oggetti grafici "Shapes" (Forme).

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Shapes" (Forme) e una descrizione dei componenti rilevanti della libreria standard MQL5.

Nome della classe	Oggetto
CChartObjectRectangle	"Rectangle" oggetto grafico "Rettangolo"
CChartObjectTriangle	"Triangle" oggetto grafico "Triangolo"
CChartObjectEllipse	"Ellipse" oggetto grafico "Ellisse"

Vedi anche

Tipi di oggetti, Oggetti grafici

CChartObjectRectangle

CChartObjectRectangle è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Rectangle".

Descrizione

La Classe CChartObjectRectangle fornisce l'accesso alle proprietà dell'oggetto "Rectangle" (Rettangolo) .

Dichiarazione

```
class CChartObjectRectangle : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Gerarchia di ereditarietà

CObject

CChartObject

CChartObjectRectangle

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Rectangle"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Rectangle".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   long      window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2         // coordinate del secondo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Tipo d'identificatore oggetto (OBJ_RECTANGLE per CChartObjectRectangle).

CChartObjectTriangle

CChartObjectTriangle è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Triangle".

Descrizione

La Classe CChartObjectTriangle fornisce l'accesso alle proprietà dell'oggetto "Triangle" (Triangolo).

Dichiarazione

```
class CChartObjectTriangle : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectTriangle

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Triangle"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Triangle".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   long      window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2,        // coordinate del secondo prezzo
   datetime  time3,         // coordinate del terzo orario
   double    price3         // coordinate del terzo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Prezzo per il terzo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Tipo d'identificatore oggetto (OBJ_TRIANGLE per CChartObjectTriangle).

CChartObjectEllipse

CChartObjectEllipse è una classe per l'accesso semplificato alle proprietà degll' oggetto grafico "Ellisse".

Descrizione

La Classe CChartObjectEllipse fornisce l'accesso alle proprietà dell'oggetto "Ellipse" (Ellisse).

Dichiarazione

```
class CChartObjectEllipse : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectEllipse

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Ellisse"
Input/output	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Vedi anche

Tipi di oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "Ellipse".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time1,         // coordinate del primo orario
   double    price1,        // coordinate del primo prezzo
   datetime  time2,         // coordinate del secondo orario
   double    price2,        // coordinate del secondo prezzo
   datetime  time3,         // coordinate del terzo orario
   double    price3         // coordinate del terzo prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time1

[in] Coordinate temporali del primo punto di ancoraggio.

price1

[in] Coordinate prezzo del primo punto di ancoraggio.

time2

[in] Coordinate temporali del secondo punto di ancoraggio.

price2

[in] Coordinate prezzo del secondo punto di ancoraggio.

time3

[in] Coordinate Temporali per il terzo punto di ancoraggio.

price3

[in] Coordinate Prezzo per il terzo punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
int  Type() const
```

Valore di ritorno

Tipo identificatore oggetto (OBJ_ELLIPSE per CChartObjectEllipse).

Oggetti Freccia

Gruppo per gli oggetti grafici Freccia (Arrows).

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Arrow" (frecce) e una descrizione dei componenti rilevanti della libreria standard MQL5. In sostanza, la freccia è una certa icona che è associata con un codice specifico. Ci sono due tipi di oggetto grafico "Freccia" per visualizzare le icone nel chart:

- Oggetto "Arrow", che consente di specificare il codice dell' icona visualizzata dall'oggetto.
- Gruppo di oggetti per visualizzare un certo tipo di icone (corrispondente ad un certo codice fisso).

Classe per lavorare con le frecce visualizzando codici di icone arbitrari

Nome della classe	Nome dell'oggetto freccia
CChartObjectArrow	Arrow

Classi lavoro con le frecce che mostrano un'icona a codice fisso

Nome della classe	Nome dell'oggetto freccia
CChartObjectArrowCheck	Check (controllo)
CChartObjectArrowDown	Arrow Up (Freccia Su)
CChartObjectArrowUp	Arrow Down (Freccia Giu)
CChartObjectArrowStop	Stop Sign (Segno di Stop)
CChartObjectArrowThumbDown	Thumbs Up (Pollice Su)
CChartObjectArrowThumbUp	Thumbs Down (Pollice Giu)
CChartObjectArrowLeftPrice	Left Price Label (Etichetta prezzo sinistra)
CChartObjectArrowRightPrice	Right Price Label (Etichetta prezzo destra)

Vedi anche

Tipi di Oggetti, Metodi per l'associazione Oggetti, Oggetti Grafici

CChartObjectArrow

CChartObjectArrow è una classe per l'accesso semplificato alla proprietà "Freccia" degli oggetti grafici.

Descrizione

La Classe CChartObjectArrow fornisce l'accesso alle proprietà comuni degli oggetti "Freccia" per tutti i suoi discendenti.

Dichiarazione

```
class CChartObjectArrow : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectArrow

Discendenti diretti

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

I Metodi della Classe per Gruppi

Create	
Create	Crea l' oggetto grafico "Arrow"
Proprietà	
ArrowCode	Ottiene/Imposta la proprietà "Codice Freccia"
Ancora(ancoraggio)	Ottiene/Imposta la proprietà "Anchor"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Vedi anche

Tipi di Oggetto, Metodi per la legatura oggetto, Oggetti Grafici

Create

Crea l'oggetto grafico "Arrow" .

```
bool  Create(
   long      chart_id,      // ID chart
   string    name,          // Nome dell'oggetto
   int       window,        // Finestra chart
   datetime  time,          // orario
   double    price,         // prezzo
   char      code           // codice freccia
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Nome dell'oggetto univoco.

window

[in] Numero finestra Chart (0 - finestra principale).

time

[in] Coordinate temporali.

price

[in] Coordinate prezzo.

code

[in] Codice "Arrow" (Wingdings).

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CChartObjectArrow::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- imposta parametri oggetto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- errore creazione freccia
      printf("Creazione Freccia: Errore %d!",GetLastError());
      //---
```

```
      return;
     }
//--- usa freccia
//--- . . .
  }
```

ArrowCode (Metodo Get)

Ottiene il codice simbolo "Arrow".

```
char  ArrowCode() const
```

Valore di ritorno

Codice simbolo "Arrow" dell'oggetto assegnato all'istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

ArrowCode (Metodo Set)

Imposta il codice simbolo per "Arrow"

```
bool  ArrowCode(
   char  code      // valore del codice
   )
```

Parametri

code

[in] Nuovo valore per il codice "arrow" (Wingdings).

Valore di ritorno

true - successo, false - non si può cambiare il codice.

Esempio:

```
//--- esempio per CChartObjectArrow::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- imposta parametri oggetto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
     {
      //--- errore creazione freccia
      printf("Creazione freccia: Errore %d!",GetLastError());
      //---
      return;
     }
//--- camvia il codice della freccia
//--- . . .
//--- ottiene il codice della freccia
   if(arrow.ArrowCode()!=code)
     {
     //--- imposta il codice della freccia
```

```
      arrow.ArrowCode(code);
      }
//--- usa freccia
//--- . . .
   }
```

Anchor (Metodo Get)

Ottiene il tipo di Ancora(ancoraggio) dell'oggetto "Arrow"

```
ENUM_ARROW_ANCHOR  Anchor() const
```

Valore di ritorno

Tipo di ancora dell' oggetto "Arrow" assegnato alla istanza della classe (al chart). Se non c'è alcun oggetto assegnato, restituisce WRONG_VALUE.

Anchor (Metodo Set)

Imposta il tipo di ancora per l'oggetto "Arrow"

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // tipo di ancora
   )
```

Parametri

anchor

[in] Nuovo valore del tipo di ancora

Valore di ritorno

true - successo, false - non posso cambiare il tipo di ancora.

Esempio:

```
//--- esempio per CChartObject::Anchor
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- imposta parametri oggetto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- errore creazione freccia
      printf("Creazione freccia: Errore %d!",GetLastError());
      //---
      return;
     }
//--- ottiene l'ancora della freccia
   if(arrow.Anchor()!=anchor)
     {
     //--- imposta l'ancora della freccia
     arrow.Anchor(anchor);
     }
```

```
//--- usa freccia
//--- . . .
  }
```

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file aperto in precedenza tramite la funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CChartObjectArrow::Save
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- imposta parametri oggetto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- errore creazione freccia
      printf("Creazione freccia: Errore %d!",GetLastError());
      //---
      return;
     }
//--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- errore salvataggio file
         printf("Salva file: Errore %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file precedentemente aperto con la funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CChartObjectArrow::Load
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- apri file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- errore caricamentoe file
         printf("Caricamento file: Errore %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- usa freccia
//--- . . .
  }
```

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Idendificatore del tipo di oggetto (ad esempio, OBJ_ARROW per CChartObjectArrow)

Esempio:

```
//--- esempio per CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- ottiene il tipo arrow
   int type=arrow.Type();
  }
```

Frecce con codice fisso

"Frecce con codice fisso" sono classi per l'accesso semplificato alle proprietà dei seguenti oggetti grafici:

Nome della classe	Nome dell'oggetto freccia
CChartObjectArrowCheck	"Arrow Check"
CChartObjectArrowDown	"Arrow Down"
CChartObjectArrowUp	"Arrow Up"
CChartObjectArrowStop	"Arrow Stop"
CChartObjectArrowThumbDown	"Good" ("Pollice su")
CChartObjectArrowThumbUp	"Bad" ("Pollice giu")
CChartObjectArrowLeftPrice	Freccia "Prezzo a sinistra"
CChartObjectArrowRightPrice	Freccia "Prezzo a destra"

Descrizione

"Frecce con codice fisso", classi che forniscono l'accesso alle proprietà dell'oggetto.

Dichiarazioni

```
class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown       : public CChartObjectArrow;
class CChartObjectArrowUp         : public CChartObjectArrow;
class CChartObjectArrowStop       : public CChartObjectArrow;
class CChartObjectArrowThumbDown  : public CChartObjectArrow;
class CChartObjectArrowThumbUp    : public CChartObjectArrow;
class CChartObjectArrowLeftPrice  : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;
```

Titolo

```
<ChartObjects\ChartObjectsArrows.mqh>
```

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico corrispondente alla classe
Proprietà	
ArrowCode	"Stub" per il metodo del cambio di codice del simbolo
Input/output	
virtual Type	Metodo virtuale di identificazione

Vedi anche

Tipi di oggetti, Metodi di legatura oggetto, Oggetti grafici

Create

Crea l'oggetto grafico "Freccia con codice fisso" .

```
bool  Create(
   long      chart_id,      // ID chart
   string    name,          // Nome dell'oggetto
   int       window,        // Finestra chart
   datetime  time,          // orario
   double    price          // prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Nome univoco dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time

[in] Coordinate temporali.

price

[in] Coordinate prezzo.

Valore di ritorno

true - successo, false - errore.

Esempio:

```
//--- esempio per CChartObjectArrowCheck::Create
//--- esempio per CChartObjectArrowDown::Create
//--- esempio per CChartObjectArrowUp::Create
//--- esempio per CChartObjectArrowStop::Create
//--- esempio per CChartObjectArrowThumbDown::Create
//--- esempio per CChartObjectArrowThumbUp::Create
//--- esempio per CChartObjectArrowLeftPrice::Create
//--- esempio per CChartObjectArrowRightPrice::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- per esempio, prendi CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- imposta parametri oggetto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

```
      {
       //--- errore crezione freccia
       printf("Creazione freccial: Errore %d!",GetLastError());
       //---
       return;
      }
//--- usa freccia
//--- . . .
  }
```

ArrowCode

Proibisce modifiche al codice simbolo "Freccia".

```
bool  ArrowCode(
   char  code      // valore del codice
   )
```

Parametri

code

[in] Qualsiasi valore

Valore di ritorno

Sempre false.

Esempio:

```
//--- esempio per CChartObjectArrowCheck::ArrowCode
//--- esempio per CChartObjectArrowDown::ArrowCode
//--- esempio per CChartObjectArrowUp::ArrowCode
//--- esempio per CChartObjectArrowStop::ArrowCode
//--- esempio per CChartObjectArrowThumbDown::ArrowCode
//--- esempio per CChartObjectArrowThumbUp::ArrowCode
//--- esempio per CChartObjectArrowLeftPrice::ArrowCode
//--- esempio per CChartObjectArrowRightPrice::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- per esempio, prendi CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- imposta parametri oggetto
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- errore creazione freccia
      printf("Creazione freccia: Errore %d!",GetLastError());
      //---
      return;
     }
//--- imposta il codice della freccia
   if(!arrow.ArrowCode(181))
     {
      //--- non è un errore
      printf("Il codice della freccia non può essere cambiato");
     }
//--- usa freccia
//--- . . .
  }
```

Type

Restituisce l'identificatore del tipo di oggetto grafico

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del Tipo d'oggetto:

CChartObjectArrowCheck - OBJ_ARROW_CHECK,

CChartObjectArrowDown - OBJ_ARROW_DOWN,

CChartObjectArrowUp - OBJ_ARROW_UP,

CChartObjectArrowStop - OBJ_ARROW_STOP,

CChartObjectArrowThumbDown - OBJ_ARROW_THUMB_DOWN,

CChartObjectArrowThumbUp - OBJ_ARROW_THUMB_UP,

CChartObjectArrowLeftPrice - OBJ_ARROW_LEFT_PRICE,

CChartObjectArrowRightPrice - OBJ_ARROW_RIGHT_PRICE.

Esempio:

```
//--- esempio per CChartObjectArrowCheck::Type
//--- esempio per CChartObjectArrowDown::Type
//--- esempio per CChartObjectArrowUp::Type
//--- esempio per CChartObjectArrowStop::Type
//--- esempio per CChartObjectArrowThumbDown::Type
//--- esempio per CChartObjectArrowThumbUp::Type
//--- esempio per CChartObjectArrowLeftPrice::Type
//--- esempio per CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- per esempio, prendi CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- ottieni il tipo di freccia
   int type=arrow.Type();
  }
```

Oggetti di controllo

Un gruppo di oggetti grafici "Object Controls" (oggetti controllo).

Questa sezione contiene i dettagli tecnici di lavoro con un gruppo di classi di oggetti grafici "Object Controls" (oggetti controllo) e una descrizione dei componenti rilevanti della libreria standard MQL5.

Nome della classe	Oggetto
CChartObjectText	"Text" (oggetto grafico "Testo")
CChartObjectLabel	"Text Label" (oggetto grafico "Etichetta Testo")
CChartObjectEdit	"Edit" (oggetto grafico "Modifica")
CChartObjectButton	"Button" (oggetto grafico "Bottone")
CChartObjectSubChart	"Chart" (oggetto grafico "Chart")
CChartObjectBitmap	"Bitmap" (oggetto grafico "Bitmap")
CChartObjectBmpLabel	"Bitmap Label" (oggetto grafico "Etichetta Bitmap")
CChartObjectRectLabel	"Rectangle Label" (oggetto grafico "Etichetta Rettangolo")

Vedi anche

Tipi di oggetti, Oggetti grafici

CChartObjectText

La classe CChartObjectText è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Testo".

Descrizione

La Classe CChartObjectText fornisce l'accesso alla proprietà "Testo" dell'oggetto.

Dichiarazione

```
class CChartObjectText : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectText

Discendenti diretti

CChartObjectLabel

I Metodi della Classe per Gruppi

Create	
Create	Create le proprietà "Text" dell'oggetto grafico
Proprietà	
Angle	Ottiene/Imposta la proprietà "Angolo"
Font	Ottiene/Imposta la proprietà "Font"
FontSize	Ottiene/Imposta la proprietà "FontSize"
Ancora(ancoraggio)	Ottiene/Imposta la proprietà "Anchor"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Le classi derivate:

- CChartObjectLabel

Vedi anche

Tipi di oggetto, Le proprietà degli oggetti, I metodi di associazione oggetto, Gli oggetti grafici

Create

Crea l'oggetto grafico "Text".

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time,          // coordinate tempo
   double    price          // coordinate prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time

[in] Coordinate orarie del punto di ancoraggio.

price

[in] Coordinate prezzo del punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

Angle (metodo Get)

Ottiene il valore della proprietà "Angle"(angolo).

```
double  Angle() const
```

Valore di ritorno

Valore della proprietà "Angle" (angolo) dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce Empty_value.

Angle (Metodo Set)

Imposta un valore per la proprietà "Angle".

```
bool  Angle(
   double  angle      // valore della proprietà
   )
```

Parametri

angle

[in] Nuovo valore per la proprietà "Angle".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Font (Metodo Get)

Ottiene il valore della proprietà "Font".

```
string  Font() const
```

Valore di ritorno

Valore della proprietà "Font" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, esso restituisce "".

Font (Metodo Set)

Imposta nuovo valore per la proprietà "Font".

```
bool  Font(
   string  font      // valore della proprietà
   )
```

Parametri

font

[in] Nuovo valore per la proprietà "Font".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

FontSize (Metodo Get)

Ottiene il valore della proprietà "Fony Size" ("Dimensione Font").

```
int  FontSize() const
```

Valore di ritorno

Valore della proprietà dell'oggetto "FontSize" assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

FontSize (Metodo Set)

Imposta il nuovo valore per la proprietà "Font Size".

```
bool  FontSize(
   int  size      // valore della proprietà
   )
```

Parametri

size

[in] Nuovo valore per la proprietà "Font Size".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Anchor (Metodo Get)

Ottiene il valore della proprietà "Anchor".

```
ENUM_ANCHOR_POINT  Anchor() const
```

Valore di ritorno

Valore della proprietà "Ancor" (ancora) dell'oggetto assegnato alla istanza della classe. Se non c'è alcun oggetto assegnato, restituisce WRONG_VALUE.

Anchor (Metodo Set)

Imposta nuovo valore per la proprietà "Anchor".

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // valore della proprietà
   )
```

Parametri

anchor

[in] Nuovo valore per la proprietà "Anchor".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Object type identifier (OBJ_TEXT for CChartObjectText).

CChartObjectLabel

CChartObjectLabel è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Label".

Descrizione

La Classe CChartObjectLabel fornisce l'accesso alle proprietà dell'oggetto "Label".

Dichiarazione

```
class CChartObjectLabel : public CChartObjectText
```

Titolo

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectText
CChartObjectLabel

Discendenti diretti

CChartObjectEdit, CChartObjectRectLabel

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Label"
Proprietà	
X_Distance	Ottiene/Imposta la proprietà "X_Distance"
Y_Distance	Ottiene/Imposta la proprietà "Y_Distance"
X_Size	Ottiene/Imposta la proprietà "X_Size"
Y_Size	Ottiene/Imposta la proprietà "Y_Size"
Corner	Ottiene/Imposta la proprietà "Corner"
Time	"Stub" per il cambio di coordinate temporali
Price	"Stub" per il cambio di coordinate prezzo
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Vedi anche

Tipi di oggetto, Le proprietà degli oggetti, Angolo chart, Metodi di associazione Oggetto, Gli oggetti grafici

Create

Crea l'oggetto grafico "Label".

```
bool  Create(
   long    chart_id,      // identificatore chart
   string  name,          // nome oggetto
   int     window,        // finestra chart
   int     X,             // X coordinate
   int     Y              // Y coordinate
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

X

[in] X coordinate.

Y

[in] Y coordinate.

Valore di ritorno

true - successo, false - errore.

X_Distance (Metodo Get)

Ottiene il valore della proprietà "X_Distance".

```
int  X_Distance() const
```

Valore di ritorno

Valore della proprietà "X_Distance" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

X_Distance (Metodo Set)

Imposta nuovo valore per la proprietà "X_Distance".

```
bool  X_Distance(
   int  X      // valore della proprietà
   )
```

Parametri

X

[in] Nuovo valore per la proprietà "X_Distance".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Y_Distance (Metodo Get)

Ottiene il valore della proprietà "Y_Distance".

```
int  Y_Distance() const
```

Valore di ritorno

Valore della proprietà "Y_Distance" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Distance (Metodo Set)

Imposta nuovo valore per la proprietà "Y_Distance".

```
bool  Y_Distance(
   int  Y      // valore della proprietà
   )
```

Parametri

Y

[in] Nuovo valore per la proprietà "Y_Distance".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

X_Size

Ottiene il valore della proprietà "X_Size".

```
int  X_Size() const
```

Valore di ritorno

Valore della proprietà "X_Size" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Size

Ottiene il valore della proprietà "Y_Size".

```
int  Y_Size() const
```

Valore di ritorno

Valore della proprietà "Y_Size" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Corner (Metodo Get)

Ottiene il valore della proprietà "Corner".

```
ENUM_BASE_CORNER  Corner() const
```

Valore di ritorno

Valore della proprietà "Corner" dell'oggetto assegnato alla istanza di classe. Se non c'è alcun oggetto assegnato, restituisce WRONG_VALUE.

Corner (Metodo Set)

Imposta nuovo valore per la proprietà "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // valore della proprietà
   )
```

Parametri

corner

[in] Nuovo valore per la proprietà "Corner".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Time

Vieta cambiamenti delle coordinate tempo.

```
bool  Time(
   datetime  time      // qualsiasi valore
   )
```

Parametri

time

[in] Qualunque valore di tipo datetime.

Valore di ritorno

sempre false.

Price

Vieta modifiche delle coordinate prezzo.

```
bool  Price(
   double  price      // qualsiasi valore
   )
```

Parametri

price

[in] Qualsiasi valore di tipo double.

Valore di ritorno

sempre false.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d'identificatore oggetto (OBJ_LABEL per CChartObjectLabel).

CChartObjectEdit

CChartObjectEdit è una classe per l'accesso semplificato alle proprietà "Modifica" dell'oggetto grafico.

Descrizione

La Classe CChartObjectEdit fornisce l'accesso alla proprietà "Modifica" dell'oggetto.

Dichiarazione

```
class CChartObjectEdit : public CChartObjectLabel
```

Titolo

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit

Discendenti diretti

CChartObjectButton

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto oggetto grafico "Modifica"
Proprietà	
TextAlign	Ottiene/Imposta la proprietà "TextAlign"
X_Size	Ottiene la proprietà "X Size"
Y_Size	Ottiene la proprietà "Y Size"
BackColor	Ottiene/Imposta la proprietà "Colore di Sfondo"
BorderColor	Ottiene/Imposta la proprietà "Colore del Bordo"
ReadOnly	Ottiene/Imposta la proprietà "Sola Lettura"
Angle	Ottiene/Imposta la proprietà "Angolo"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file

Create	
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Metodi ereditati dalla classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Vedi anche

Tipi di oggetto, Le proprietà degli oggetti, Angolo chart, Metodi di associazione Oggetto, Gli oggetti grafici

Create

Crea l'oggeto grafico "Edit" .

```
bool  Create(
   long    chart_id,      // identificatore chart
   string  name,          // nome oggetto
   int     window,        // finestra chart
   int     X,             // X coordinate
   int     Y,             // Y coordinate
   int     sizeX,         // grandezza X
   int     sizeY          // grandezza Y
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

X

[in] X coordinate.

Y

[in] Y coordinate.

sizeX

[in] grandezza X.

sizeY

[in] grandezza Y.

Valore di ritorno

true - successo, false - errore.

TextAlign (Metodo Get)

Ottiene il valore della proprietà "TextAlign" (modalità allineamento del testo).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valore di ritorno

Valore della proprietà "TextAlign" dell'oggetto assegnato alla istanza della classe.

TextAlign (Set method)

Imposta il valore della proprietà "TextAlign" (modalità allineamento del testo).

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // valore della proprietà
   )
```

Parametri

align

[in] Nuovo valore della proprietà "TextAlign".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

X_Size

Imposta il valore per la proprietà "X_Size".

```
bool  X_Size(
   int  size      // valore della proprietà
   )
```

Parametri

size

[in] Nuovo valore per la proprietà "X_Size".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

Y_Size

Imposta il valore per la proprietà "Y_Size".

```
bool  Y_Size(
   int  size      // valore della proprietà
   )
```

Parametri

size

[in] Nuovo valore per la proprietà "Y_Size".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

BackColor (Metodo Get)

Ottiene il valore della proprietà "BackColor".

```
color  BackColor() const
```

Valore di ritorno

Valore della proprietà "BackColor" dell'oggetto assegnato all' istanza della classe. Se non c'è un oggetto assegnato, restituisce CLR_NONE.

BackColor (Metodo Set)

Imposta il valore per la proprietà "BackColor".

```
bool  BackColor(
   color  new_color      // valore della proprietà
   )
```

Parametri

new_color

[in] Nuovo valore per la proprietà "BackColor".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

BorderColor (Metoto Get)

Ottiene il valore della proprietà "Colore del bordo".

```
color  BorderColor() const
```

Valore di ritorno

Valore della proprietà "Colore del bordo" dell'oggetto assegnato alla istanza della classe. Se non c'è un oggetto assegnato, restituisce CLR_NONE.

BorderColor (Metoto Set)

Imposta nuovo valore per la proprietà "Colore del bordo".

```
bool  BorderColor(
   color  new_color      // nuovo valore della proprietà
   )
```

Parametri

new_color

[in] Nuovo valore per la proprietà "Border Color".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

ReadOnly (Metodo Get)

Ottiene il valore della proprietà "Sola lettura".

```
bool  ReadOnly() const
```

Valore di ritorno

Valore della proprietà "Sola lettura" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

ReadOnly (Metodo Set)

Imposta il valore della proprietà "Sola lettura".

```
bool  ReadOnly(
   const  bool  flag      // nuovo valore della proprietà
   )
```

Parametri

flag

[in] Nuovo valore per proprietà "Sola Lettura" (true significa che l'editing del testo è disabilitato).

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

Angle

Vieta modifiche della proprietà "Angolo".

```
bool  Angle(
   double  angle      // qualsiasi valore
   )
```

Parametri

angle

[in] Qualsiasi valore di tipo double.

Valore di ritorno

sempre false.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d'identificatore oggetto (OBJ_EDIT per CChartObjectEdit).

CChartObjectButton

CChartObjectButton è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Button".

Descrizione

La Classe CChartObjectButton fornisce l'accesso alle proprietà dell'oggetto "Button".

Dichiarazione

```
class CChartObjectButton : public CChartObjectEdit
```

Titolo

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit
CChartObjectButton

Discendenti diretti

CChartObjectPanel

I Metodi della Classe per Gruppi

Create	
Create	Ereditato dalla classe CChartObjectEdit
Proprietà	
State	Ottiene/Imposta lo stato del bottone (Premuto/Rilasciato)
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Metodi ereditati dalla classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Metodi ereditati dalla classe CChartObjectEdit

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

Vedi anche

Tipi di oggetti, Proprietà oggetti, Angolo Chart, Metodi di associazione oggetti, Oggetti grafici

State (Metodo Get)

Ottiene il valore della proprietà "State".

```
bool  State() const
```

Valore di ritorno

Valore della proprietà "Stato" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

State (Metodo Set)

Imposta nuovo valore per la proprietà "State".

```
bool  State(
   bool  state      // valore della proprietà
   )
```

Parametri

X

[in] Nuovo valore per la proprietà "State".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di identificatore oggetto (OBJ_BUTTON per CChartObjectButton).

CChartObjectSubChart

CChartObjectSubChart è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Chart".

Descrizione

La Classe CChartObjectSubChart fornisce l'accesso alle proprietà dell'oggetto "Chart".

Dichiarazione

```
class CChartObjectSubChart : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectSubChart

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Chart"
Proprietà	
X_Distance	Ottiene/Imposta la proprietà "X_Distance"
Y_Distance	Ottiene/Imposta la proprietà "Y_Distance"
Corner	Ottiene/Imposta la proprietà "Corner"
X_Size	Ottiene/Imposta la proprietà "X_Size"
Y_Size	Ottiene/Imposta la proprietà "Y_Size"
Symbol	Ottiene/Imposta la proprietà "Simbolo"
Period	Ottiene/Imposta la proprietà "Periodo"
Scale	Ottiene/Imposta la proprietà "Scale"
DateScale	Ottiene/Imposta la proprietà "Mostra data della scala"
PriceScale	Ottiene/Imposta la proprietà "Mostra scala prezzo"
Time	"Stub" per il cambio di coordinate temporali
Price	"Stub" per il cambio di coordinate prezzo
Input/output	

Create	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Vedi anche

Tipi di oggetti, Proprietà oggetti, Angolo Chart, Oggetti grafici

Create

Crea l'oggetto grafico "SubChart".

```
bool  Create(
   long     chart_id,      // identificatore chart
   string   name,          // nome oggetto
   int      window,        // finestra chart
   int      X,             // X coordinate
   int      Y,             // Y coordinate
   int      sizeX,         // grandezza X
   int      sizeY          // grandezza Y
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

X

[in] X coordinate.

Y

[in] Y coordinate.

sizeX

[in] grandezza X.

sizeY

[in] grandezza Y.

Valore di ritorno

true - successo, false - errore.

X_Distance (Metodo Get)

Ottiene il valore della proprietà "X_Distance".

```
int  X_Distance() const
```

Valore di ritorno

Valore della proprietà "X_Distance" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

X_Distance (Metodo Set)

Imposta nuovo valore per la proprietà "X_Distance".

```
bool  X_Distance(
   int  X      // valore della proprietà
   )
```

Parametri

X

[in] Nuovo valore per la proprietà "X_Distance".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Y_Distance (Metodo Get)

Ottiene il valore della proprietà "Y_Distance".

```
int  Y_Distance() const
```

Valore di ritorno

Valore della proprietà "Y_Distance" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Distance (Metodo Set)

Imposta nuovo valore per la proprietà "Y_Distance".

```
bool  Y_Distance(
   int  Y      // valore della proprietà
   )
```

Parametri

Y

[in] Nuovo valore per la proprietà "Y_Distance".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Corner (Metodo Get)

Ottiene il valore della proprietà "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valore di ritorno

Valore della proprietà "Corner" dell'oggetto assegnato alla istanza di classe. Se non c'è alcun oggetto assegnato, restituisce WRONG_VALUE.

Corner (Metodo Set)

Imposta nuovo valore per la proprietà "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // valore della proprietà
   )
```

Parametri

corner

[in] Nuovo valore per la proprietà "Corner".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

X_Size (Metodo Get)

Ottiene il valore della proprietà "X_Size".

```
int  X_Size() const
```

Valore di ritorno

Valore della proprietà "X_Size" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

X_Size (Metodo Set)

Imposta nuovo valore per la proprietà "X_Size".

```
bool  X_Size(
   int  X      // valore della proprietà
   )
```

Parametri

X

[in] Nuovo valore per la proprietà "X_Size".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Y_Size (Metodo Get)

Ottiene il valore della proprietà "Y_Size".

```
int  Y_Size() const
```

Valore di ritorno

Valore della proprietà "Y_Size" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Size (Metodo Set)

Imposta nuovo valore per la proprietà "Y_Size".

```
bool  Y_Size(
   int  Y      // valore della proprietà
   )
```

Parametri

Y

[in] Nuovo valore per la proprietà "Y_Size".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Symbol (Metodo Get)

Ottiene il valore della proprietà "Simbolo".

```
string  Symbol() const
```

Valore di ritorno

Valore della proprietà "Symbol" dell'oggetto assegnato all' istanza della classe. Se non vi è alcun oggetto assegnato, esso restituisce "".

Symbol (Metodo Set)

Imposta nuovo valore per la proprietà "Symbol".

```
bool  Symbol(
   string  symbol      // simbolo
   )
```

Parametri

symbol

[in] Nuovo valore per la proprietà "Simbolo".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Period (Metodo Get)

Ottiene il valore della proprietà "Periodo".

```
int  Period() const
```

Valore di ritorno

Valore della proprietà "Periodo" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Period (Metodo Set)

Imposta nuovo valore per la proprietà "Period".

```
bool  Period(
   int  period      // periodo
   )
```

Parametri

period

[in] Nuovo valore per la proprietà "Period".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Scale (Metodo Get)

Ottiene il valore della proprietà "Scala".

```
double  Scale() const
```

Valore di ritorno

Valore per la proprietà "Scale" dell'oggetto assegnato all'istanza della classe. Se non c'è un oggetto assegnato, restituisce EMPTY_VALUE.

Scale (Metodo Set)

Imposta nuovo valore per la proprietà "Scale".

```
bool  Scale(
   double  scale      // valore della proprietà
   )
```

Parametri

scale

[in] Nuovo valore per la proprietà "Scale".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

DateScale (Metodo Get)

Ottiene il valore della flag "DateScale".

```
bool  DateScale() const
```

Valore di ritorno

Valore della flag "DateScale" dell'oggetto assegnato all'istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

DateScale (Metodo Set)

Imposta nuovo valore per la proprietà "DateScale".

```
bool  DateScale(
   bool  scale      // valore del flag
   )
```

Parametri

scale

[in] Nuovo valore per la flag"DateScale".

Valore di ritorno

true - successo, false - non posso cambiare il flag.

PriceScale (Metodo Get)

Ottiene il valore del flag "PriceScale".

```
bool  PriceScale() const
```

Valore di ritorno

Valore della flag "PriceScale" dell'oggetto assegnato all'istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

PriceScale (Metodo Set)

Imposta nuovo valore per la flag "PriceScale".

```
bool  PriceScale(
   bool  scale      // valore del flag
   )
```

Parametri

scale

[in] Nuovo valore per la flag "PriceScale".

Valore di ritorno

true - successo, false - non posso cambiare il flag.

Time

Vieta cambiamenti delle coordinate tempo.

```
bool  Time(
   datetime  time      // qualsiasi valore
   )
```

Parametri

time

[in]

Valore di ritorno

sempre false.

Price

Vieta modifiche delle coordinate prezzo.

```
bool  Price(
   double  price      // qualsiasi valore
   )
```

Parametri

price

[in]

Valore di ritorno

sempre false.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d'identificatore oggetto (OBJ_CHART CChartObjectSubChart).

CChartObjectBitmap

La Classe CChartObjectBitmap è una classe per l'accesso semplificato alle proprietà dell'oggetto grafico "Bitmap"

Descrizione

La Classe CChartObjectBitmap fornisce l'accesso alle proprietà dell'oggetto "Bitmap".

Dichiarazione

```
class CChartObjectBitmap : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectBitmap

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "Bitmap"
Proprietà	
BmpFile	Ottiene/Imposta la proprietà "BMP Filename"
X_Offset	Ottiene/Imposta la proprietà "X_Offset"
Y_Offset	Ottiene/Imposta la proprietà "Y_Offset"
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Vedi anche

Tipi di oggetti, Proprietà oggetti, Oggetti grafici

Create

Creates "Bitmap" graphical object.

```
bool  Create(
   long      chart_id,      // identificatore chart
   string    name,          // nome dell'oggetto
   int       window,        // finestra chart
   datetime  time,          // coordinate tempo
   double    price          // coordinate prezzo
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

time

[in] Coordinate orarie del punto di ancoraggio.

price

[in] Coordinate prezzo del punto di ancoraggio.

Valore di ritorno

true - successo, false - errore.

BmpFile (Metodo Get)

Ottiene il valore della proprietà "BmpFile".

```
string  BmpFile() const
```

Valore di ritorno

Valore della proprietà dell'oggetto "BmpFile" assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

BmpFile (Metodo Set)

Imposta nuovo valore per la proprietà "BmpFile".

```
bool  BmpFile(
   string  name      // valore della proprietà
   )
```

Parametri

name

[in] Nuovo valore per la proprietà "BmpFile".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

X_Offset (Metodo Get)

Ottiene il valore della proprietà "X_Offset" (in alto a sinistra) dell'oggetto grafico CChartObjectBitmap.

```
int  X_Offset() const
```

Valore di ritorno

Valore della proprietà dell'oggetto "X_offset" assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

X_Offset (Metodo Set)

Imposta nuovo valore per la proprietà "X_offset" (in alto a sinistra) dell'oggetto grafico CChartObjectBitmap. Il valore è impostato in pixel relativi all'angolo superiore sinistro dell'immagine originale.

```
bool  X_Offset(
   int  X      // valore della proprietà
   )
```

Parametri

X

[in] Nuovo valore per la proprietà "X_Offset" .

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Y_Offset (Metodo Get)

Ottiene il valore della proprietà "Y_Offset" (in alto a sinistra) dell'oggetto grafico CChartObjectBitmap.

```
int  Y_Offset() const
```

Valore di ritorno

Valore della proprietà "Y_Offset" dell'oggetto assegnato alla istanza dell' classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Offset (Metodo Set)

Imposta nuovo valore per la proprietà "Y_Offset" (in alto a sinistra) dell'oggetto grafico CChartObjectBitmap. Il valore è impostato in pixel relativi all'angolo superiore sinistro dell'immagine originale.

```
bool  Y_Offset(
   int  Y      // valore della proprietà
   )
```

Parametri

Y

[in] Nuovo valore per la proprietà "Y_Offset".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del topo d'oggetto (OBJ_BITMAP per CChartObjectBitmap).

CChartObjectBmpLabel

CChartObjectBmpLabel è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Bitmal Label".

Descrizione

La Classe CChartObjectBmpLabel fornisce l'accesso alle proprietà dell'oggetto "Bitmap Label".

Dichiarazione

```
class CChartObjectBmpLabel : public CChartObject
```

Titolo

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Gerarchia di ereditarietà

CObject
  CChartObject
    CChartObjectBmpLabel

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "BmpLabel"
Proprietà	
X_Distance	Ottiene/Imposta la proprietà "X_Distance"
Y_Distance	Ottiene/Imposta la proprietà "Y_Distance"
X_Offset	Ottiene/Imposta la proprietà "X_Offset"
Y_Offset	Ottiene/Imposta la proprietà "Y_Offset"
Corner	Ottiene/Imposta la proprietà "Corner"
X_Size	Ottiene/Imposta la proprietà "X_Size"
Y_Size	Ottiene/Imposta la proprietà "Y_Size"
BmpFileOn	Ottiene/Imposta la proprietà "BmpFileOn" per lo stato di bottone premuto (On)
BmpFileOff	Ottiene/Imposta la proprietà "BmpFileOff" per stato di bottone rilasciato (Off)
State	Ottiene/Imposta "lo stato del bottone" (Premuto/Rilasciato)
Time	"Stub" per il cambio di coordinate temporali
Price	"Stub" per il cambio di coordinate prezzo

Create	
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Vedi anche

Tipi di oggetti, Proprietà oggetti, Angolo Chart, Oggetti grafici

Create

Crea l'oggetto grafico "BmpLabel".

```
bool  Create(
   long    chart_id,      // identificatore chart
   string  name,          // nome oggetto
   int     window,        // finestra chart
   int     X,             // X coordinate
   int     Y              // Y coordinate
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

X

[in] X coordinate.

Y

[in] Y coordinate.

Valore di ritorno

true - successo, false - errore.

X_Distance (Metodo Get)

Ottiene il valore della proprietà "X_Distance".

```
int  X_Distance() const
```

Valore di ritorno

Valore della proprietà "X_Distance" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

X_Distance (Metodo Set)

Imposta nuovo valore per la proprietà "X_Distance".

```
bool  X_Distance(
   int  X      // valore della proprietà
   )
```

Parametri

X

[in] Nuovo valore per la proprietà "X_Distance".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Y_Distance (Metodo Get)

Ottiene il valore della proprietà "Y_Distance".

```
int  Y_Distance() const
```

Valore di ritorno

Valore della proprietà "Y_Distance" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Distance (Metodo Set)

Imposta nuovo valore per la proprietà "Y_Distance".

```
bool  Y_Distance(
   int  Y      // valore della proprietà
   )
```

Parametri

Y

[in] Nuovo valore per la proprietà "Y_Distance".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

X_Offset (Metodo Get)

Ottiene il valore della proprietà "X_Offset" (in alto a sinistra) dell'oggetto grafico CCartObjectBmpLabel.

```
int  X_Offset() const
```

Valore di ritorno

Valore della proprietà dell'oggetto "X_offset" assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

X_Offset (Metodo Set)

Imposta nuovo valore per la proprietà "X_offset" (in alto a sinistra) dell'oggetto grafico CChartObjectBitmap. Il valore è impostato in pixel relativi all'angolo superiore sinistro dell'immagine originale.

```
bool  X_Offset(
   int  X      // valore della proprietà
   )
```

Parametri

X

[in] Nuovo valore per la proprietà "X_Offset" .

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Y_Offset (Metodo Get)

Ottiene il valore della proprietà "Y_Offset" (in alto a sinistra) dell'oggetto grafico CCartObjectBmpLabel.

```
int  Y_Offset() const
```

Valore di ritorno

Valore della proprietà "Y_Offset" dell'oggetto assegnato alla istanza dell' classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Offset (Metodo Set)

Imposta nuovo valore per la proprietà "Y_Offset" (in alto a sinistra) dell'oggetto grafico CCartObjectBmpLabel. Il valore è impostato in pixel relativi all'angolo superiore sinistro dell'immagine originale.

```
bool  Y_Offset(
   int  Y      // valore della proprietà
   )
```

Parametri

Y

[in] Nuovo valore per la proprietà "Y_Offset".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Corner (Metodo Get)

Ottiene il valore della proprietà "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valore di ritorno

Valore della proprietà "Corner" dell'oggetto assegnato alla istanza di classe. Se non c'è alcun oggetto assegnato, restituisce WRONG_VALUE.

Corner (Metodo Set)

Imposta nuovo valore per la proprietà "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // valore della proprietà
   )
```

Parametri

corner

[in] Nuovo valore per la proprietà "Corner".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

X_Size

Ottiene il valore della proprietà "X_Size".

```
int  X_Size() const
```

Valore di ritorno

Valore della proprietà "X_Size" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

Y_Size

Ottiene il valore della proprietà "Y_Size".

```
int  Y_Size() const
```

Valore di ritorno

Valore della proprietà "Y_Size" dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

BmpFileOn (Metodo Get)

Ottiene il valore della proprietà "BmpFileOn".

```
string  BmpFileOn() const
```

Valore di ritorno

Valore della proprietà "BmpFileOn" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, esso restituisce "".

BmpFileOn (Metodo Set)

Imposta nuovo valore per la proprietà "BmpFileOn".

```
bool  BmpFileOn(
   string  name      // nome file
   )
```

Parametri

name

[in] Nuovo valore per la proprietà "BmpFileOn".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

BmpFileOff (Metodo Get)

Ottiene il valore della proprietà "BmpFileOff".

```
string  BmpFileOff() const
```

Valore di ritorno

Valore della proprietà "BmpFileOff" dell'oggetto assegnato alla istanza di classe. Se non vi è alcun oggetto assegnato, esso restituisce "".

BmpFileOff (Metodo Set)

Imposta nuovo valore per la proprietà "BmpFileOff".

```
bool  BmpFileOff(
   string  name      // nome file
   )
```

Parametri

name

[in] Nuovo valore per la proprietà "BmpFileOff".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

State (Metodo Get)

Ottiene il valore della proprietà "State".

```
bool  State() const
```

Valore di ritorno

Valore della proprietà "Stato" dell'oggetto assegnato alla istanza della classe. Se non vi è alcun oggetto assegnato, restituisce false.

State (Metodo Set)

Imposta nuovo valore per la proprietà "State".

```
bool  State(
   bool  state      // valore della proprietà
   )
```

Parametri

state

[in] Nuovo valore per la proprietà "State".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Time

Vieta cambiamenti delle coordinate tempo.

```
bool  Time(
   datetime  time      // qualsiasi valore
   )
```

Parametri

time

[in] Qualunque valore di tipo datetime.

Valore di ritorno

sempre false.

Price

Vieta modifiche delle coordinate prezzo.

```
bool  Price(
   double  price      // qualsiasi valore
   )
```

Parametri

price

[in] Qualsiasi valore di tipo double.

Valore di ritorno

sempre false.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Identificatore del tipo d'ggetto (OBJ_BITMAP_LABEL per CChartObjectBmpLabel).

CChartObjectRectLabel

CChartObjectRectLabel è una classe per l'accesso semplificato alla proprietà dell'oggetto grafico "Etichetta Rettangolo".

Descrizione

La Classe CChartObjectBmpLabel fornisce l'accesso alle proprietà dell'oggetto "Rettangolo Label".

Dichiarazione

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Titolo

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Gerarchia di ereditarietà

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectRectLabel

I Metodi della Classe per Gruppi

Create	
Create	Crea l'oggetto grafico "RectLabel"
Proprietà	
X_Size	Imposta la dimensione orizzontale
Y_Size	Imposta la dimensione verticale
BackColor	Ottiene/Imposta il colore di sfondo
Angle	Vieta il cambiamento di pendenza
BorderType	Ottiene/Imposta il tipo di bordo
Input/output	
virtual Save	Metodo virtuale per la scrittura su file
virtual Load	Metodo virtuale per la lettura da file
virtual Type	Metodo virtuale di identificazione

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Metodi ereditati dalla classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Metodi ereditati dalla classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Vedi anche

Tipi di oggetti, Proprietà oggetti, Oggetti grafici

Create

Crea l'oggetto grafico "CChartObjectRectLabel".

```
bool  Create(
   long     chart_id,      // ID del chart
   string   name,          // nome oggetto
   int      window,        // finstra chart
   int      X,             // X coordinate
   int      Y,             // Y coordinate
   int      sizeX,         // grandezza orizzontale
   int      sizeY          // grandezza verticale
   )
```

Parametri

chart_id

[in] Identificatore Chart (0 - chart corrente).

name

[in] Un nome unico dell'oggetto da creare.

window

[in] Numero finestra Chart (0 - finestra principale).

X

[in] X coordinate.

Y

[in] Y coordinate.

sizeX

[in] Grandezza orizzontale.

sizeY

[in] Grandezza verticale.

Valore di ritorno

true - successo, false - errore.

X_Size

Imposta il valore della proprietà "X_Size".

```
bool  X_Size(
   int  size      // valore della proprietà
   )
```

Parametri

size

[in] Nuovo valore della proprietà grandezza orizzontale.

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Nota

Per ottenere i valori delle proprietà di "X_Size" ed "Y_Size", usa i metodi X_Size e Y_Size della classe partente CChartObjectLabel.

Y_Size

Imposta il valore della proprietà "Y_Size".

```
bool  Y_Size(
   int  size      // valore della proprietà
   )
```

Parametri

size

[in] Nuovo valore della proprietà della grandezza verticale.

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Nota

Per ottenere i valori delle proprietà di "X_Size" ed "Y_Size", usa i metodi X_Size e Y_Size della classe partente CChartObjectLabel.

BackColor

Ottiene il valore della proprietà colore di sfondo.

```
color  BackColor() const
```

Valore di ritorno

Valore della proprietà Colore di sfondo dell'oggetto assegnato alla istanza della classe. Se non c'è oggetto assegnato, restituisce 0.

BackColor

Imposta il valore della proprietà colore di sfondo.

```
bool  BackColor(
   color  new_color      // valore della proprietà
   )
```

Parametri

new_color

[in] Nuovo valore della proprietà colore di sfondo.

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Angle

Vieta la modifica della proprietà angolo di inclinazione.

```
bool  Angle(
   double  angle      // qualsiasi valore
   )
```

Parametri

angle

[in] Qualsiasi valore di tipo double.

Valore di ritorno

Sempre false.

BorderType

Ottiene il valore della proprietà tipo di bordo.

```
int  BorderType() const
```

Valore di ritorno

Valore della proprietà tipo di bordo dell'oggetto assegnato all'istanza della classe. Se non vi è alcun oggetto assegnato, restituisce 0.

BorderType

Consente di impostare il valore della proprietà tipo di bordo.

```
bool  BorderType(
   int  type      // valore della proprietà
   )
```

Parametri

type

[in] Nuovo tipo di bordo (valore della proprietà).

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] handle del file binario già aperto dalla funzione FileOpen.

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore dell'oggetto grafico.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d'identificatore oggetto (OBJ_RECTANGLE_LABEL per CChartObjectRectangleLabel).

Grafica personalizzata

Questa sezione fornisce gli strumenti per lavorare con la grafica personalizzata.

Il loro utilizzo consente notevolmente la compilazione di grafici personalizzati, disegni e visualizzazione dei dati.

Ci sono classi individuali per la creazione di oggetti grafici e primitivi, per il disegno di vari tipi di grafici a torta, e curve. Sono implementate diverse opzioni per la visualizzazione di oggetti: modificare lo stile e il colore delle righe, riempire, lavorare con una serie di dati sul grafico, ecc.

Classe	Descrizione
CCanvas	Classe per la creazione semplificata di immagini personalizzate
CChartCanvas	Classe base per implementare classi destinate a disegnare grafici e loro elementi
CHistogramChart	Classe per l'analisi di istogrammi
CLineChart	Classe per curve di disegno
CPieChart	Classe per la compilazione di grafici a torta

CCanvas

CCanvas è una classe per la creazione semplificata di immagini personalizzate.

Descrizione

La Classe CCanvas fornisce la creazione di una risorsa grafica (con o senza il legarsi ad un oggetto grafico) ed il disegno di primitive grafiche.

Dichiarazione

```
class CCanvas
```

Titolo

```
#include <Canvas\Canvas.mqh>
```

Gerarchia di ereditarietà

CCanvas

Discendenti diretti

CChartCanvas, CFlameCanvas

I Metodi della Classe per Gruppi

Creazione	
Attach	Inserisce l'oggetto OBJ_BITMAP_LABEL in un'istanza della classe CCanvas
Create	Crea una risorsa grafica senza legarsi ad un oggetto del chart
CreateBitmap	Crea una risorsa grafica legata ad un oggetto del chart
CreateBitmapLabel	Crea una risorsa grafica legata ad un oggetto del chart
Destroy	Distrugge una risorsa grafica
Proprietà	
ChartObjectName	Riceve il nome di un oggetto legato al chart
ResourceName	Riceve il nome di una risorsa grafica
Larghezza	Riceve la larghezza di una risorsa grafica
Altezza	Riceve l'altezza di una risorsa grafica
LineStyleSet	Imposta lo stile della linea
Aggiornamento di un oggetto sullo schermo	
Aggiorna	Consente di visualizzare le modifiche sullo schermo

Creazione	
Ridimensiona	Ridimensiona una risorsa grafica
Cancellazione/Riempimento con il colore	
Elimina	Cancella o riempie con il colore specificato
Accesso ai dati	
PixelGet	Riceve il colore del punto con le coordinate specificate
PixelSet	Imposta il colore del punto con le coordinate specificate
Disegna primitive	
LineVertical	Disegna una linea verticale
LineHorizontal	Disegna una linea orizzontale
Line	Disegna una linea a mano libera
Polyline	Disegna una polilinea
Polygon	Disegna un poligono
Rectangle	Disegna un rettangolo
Circle	Disegna un cerchio
Triangle	Disegna un triangolo
Arc	Disegna un arco di ellisse
Pie	Disegna un settore di ellisse
Disegna primitive riempite	
FillRectangle	Disegna un rettangolo pieno
FillCircle	Disegna un cerchio pieno
FillTriangle	Disegna un triangolo pieno
FillPolygon	Disegna un poligono pieno
FillEllipse	Disegna un'ellisse piena
Fill	Riempie un'area
Disegna Primitive con antialiasing	
PixelSetAA	Disegna un pixel
LineAA	Disegna una linea
PolylineAA	Disegna una polilinea

Creazione	
PolygonAA	Disegna un poligono
TriangleAA	Disegna un triangolo
CircleAA	Disegna un cerchio
EllipseAA	Disegna un'ellisse
LineWu	Disegna una linea
PolylineWu	Disegna una polilinea
PolygonWu	Disegna un poligono
TriangleWu	Disegna un triangolo
CircleWu	Disegna un cerchio
EllipseWu	Disegna un'ellisse
Testo	
FontSet	Imposta i parametri del carattere
FontNameSet	Imposta il nome del font
FontSizeSet	Imposta font size
FontFlagsSet	Imposta bandiere dei font
FontAngleSet	Imposta angolo di pendenza del font
FontGet	Riceve parametri del font
FontNameGet	Riceve nome del font
FontSizeGet	Riceve la grandezza del font
FontFlagsGet	Riceve il flag del font
FontAngleGet	Riceve l'angolo di pendenza del font
TextOut	Visualizza il testo
TextWidth	Riceve la larghezza del testo
TextHeight	Riceve l'altezza del testo
TextSize	Riceve le dimensioni del testo
Trasparenza	
TransparentLevelSet	Imposta il livello di trasparenza
Input/output	
LoadFromFile	Legge un'immagine da un file BMP

Attach

Ottiene la risorsa grafica da un OBJ_BITMAP_LABEL oggetto e lo attribuisce ad un'istanza della classe CCanvas.

```
bool  Attach(
   const long          chart_id,                          // identificatore chart
   const string        objname,                           // nome oggetto
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // metodo elaborazione c
  );
```

Crea una risorsa grafica per un oggetto OBJ_BITMAP_LABEL e lo allega ad un'istanza della classe CCanvas.

```
bool  Attach(
   const long          chart_id,                          // identificatore chart
   const string        objname,                           // nome oggetto
   const int           width,                             // lunghezza immagine in
   const int           height,                            // altezza immagine in p
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // metodo elaborazione c
  );
```

Parametri

chart_id

[out] Identificatore del chart.

objname

[in] Nome dell'oggetto grafico.

width

[in] Larghezza dell' immagine nella risorsa.

height

[in] Altezza immagine nella risorsa.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Metodo di elaborazione del canale Alpha. Il canale alfa viene ignorato per impostazione predefinita.

Valore di ritorno

true - successo, false - se non è stato possibile allegare l'oggetto.

Arc

Disegna un arco di ellisse inscritto in un rettangolo con angoli a (x1, y1) e (x2, y2). I limiti dell'arco sono tagliati da linee dal centro dell'ellisse, che si estendono a due punti con le coordinate (x3, y3) e (x4, y4).

```
void  Arc(
   int          x1,      // coordinate X dell' angolo superiore sinistro del rettangolo
   int          y1,      // coordinate Y dell' angolo superiore sinistro del rettangolo
   int          x2,      // coordinate X dell' angolo inferiore destro del rettangolo
   int          y2,      // coordinate Y dell' angolo superiore sinistro del rettangolo
   int          x3,      // coordinate X del primo punto per trovare i limiti dell'arco
   int          y3,      // coordinate Y del primo punto per trovare i limiti dell'arco
   int          x4,      // coordinate X del secondo punto per trovare i limiti dell'ar
   int          y4,      // coordinate Y del secondo punto per trovare i limiti dell'ar
   const uint   clr      // color
   );
```

Parametri

x1

[in] coordinata X dell'angolo superiore sinistro che forma il rettangolo.

y1

[in] coordinata Y dell'angolo superiore sinistro che forma il rettangolo.

x2

[in] coordinata X dell'angolo in basso a destra che forma il rettangolo.

y2

[in] coordinata Y dell'angolo in basso a destra che forma il rettangolo.

x3

[in] coordinata X del primo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

y3

[in] coordinata Y del primo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

x4

[in] coordinata X del secondo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

y4

[in] coordinata Y del secondo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

clr

[in] Colore in formato ARGB. Usa la funzione ColorToARGB() per convertire un colore nel formato ARGB.

Disegna un arco di ellisse con centro al punto (x, y), inciso in rettangolo, con raggi x e ry. I limiti dell'arco vengono ritagliati dal centro dell'ellisse utilizzando i raggi formati da angoli fi3 e fi4.

```
void  Arc(
   int          x,        // coordinate X del centro ellisse
   int          y,        // coordinate Y del centro ellisse
   int          rx,       // raggio di ellisse sull'asse X
   int          ry,       // raggio di ellisse sull'asse Y
   int          fi3,      // angolo di raggio dal centro di ellisse, che definisce il pr
   int          fi4,      // angolo di raggio dal centro di ellisse, che definisce il se
   const uint   clr       // color
   );
```

Disegna un arco di ellisse con centro al punto (x, y), inciso in rettangolo, con raggi rx e ry e restituisce anche le coordinate dei limiti dell'arco. I limiti dell'arco vengono ritagliati dal centro dell'ellisse utilizzando i raggi formati da angoli fi3 e fi4.

```
void  Arc(
   int          x,        // coordinate X del centro ellisse
   int          y,        // coordinate Y del centro ellisse
   int          rx,       // raggio di ellisse sull'asse X
   int          ry,       // raggio di ellisse sull'asse Y
   int          fi3,      // angolo di raggio dal centro di ellisse, che definisce il pr
   int          fi4,      // angolo di raggio dal centro di ellisse, che definisce il se
   int&         x3,       // coordinate X del primo contorno d'arco
   int&         y3,       // coordinate Y del primo contorno d'arco
   int&         x4,       // coordinate X del secondo contorno d'arco
   int&         y4,       // coordinate Y del secondo contorno d'arco
   const uint   clr       // color
   );
```

Parametri

x

[in] coordinate X del centro ellisse.

y

[in] coordinate Y del centro ellisse.

rx

[in] Raggio dell'ellisse sull'asse Y, in pixel.

ry

[in] Raggio dell'ellisse sull'asse Y, in pixel.

fi3

[in] Angolo in radianti, che definisce il primo limite dell'arco.

fi4

[in] Angolo in radianti, che definisce il secondo limite dell'arco.

x3

[out] Variabile per ottenere la coordinata X del primo limite dell'arco.

y3

[out] Variabile per ottenere la coordinata Y del primo limite dell'arco.

x4

[out] Variabile per ottenere la coordinata X del secondo limite dell'arco.

y4

[out] Variabile per ottenere la coordinata Y del secondo limite dell'arco.

clr

[in] Colore in formato ARGB. Usa la funzione ColorToARGB() per convertire un colore nel formato ARGB.

Esempi di chiamata dei metodi della classe:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Funzione Script Programma Script                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- crea canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- pulisce canvas
   canvas.Erase(clrWhite);
//--- disegna rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- disegna primo arco
   canvas.Arc(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrRed));
   int x1,y1,x2,y2;
//--- disegna secondo arco
   canvas.Arc(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,x1,y1,x2,y2,ColorToARGB(cl
//--- stampa coordinate per arco
   PrintFormat("First point of arc at (%G,%G), second point of arc at (%G,%G)",x1,y1,x
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- mostra canvas aggiornato
   canvas.Update();
  }
```

Pie

Disegna un settore riempito di un'ellisse iscritta in un rettangolo con angoli a (x1, y1) e (x2, y2). I confini del settore vengono tagliati da linee dal centro dell'ellisse, che si estendono a due punti con le coordinate (x3, y3) e (x4, y4).

```
void  Pie(
   int          x1,      // coordinate X dell' angolo superiore sinistro del rettangolo
   int          y1,      // coordinate Y dell' angolo superiore sinistro del rettangolo
   int          x2,      // coordinate X dell' angolo inferiore destro del rettangolo
   int          y2,      // coordinate Y dell' angolo superiore sinistro del rettangolo
   int          x3,      // coordinate X del primo punto per trovare i limiti dell'arco
   int          y3,      // coordinate Y del primo punto per trovare i limiti dell'arco
   int          x4,      // coordinate X del secondo punto per trovare i limiti dell'ar
   int          y4,      // coordinate Y del secondo punto per trovare i limiti dell'ar
   const uint   clr,     // colore linea
   const uint   fill_clr // colore riempimento
   );
```

Parametri

x1

[in] coordinata X dell'angolo superiore sinistro che forma il rettangolo.

y1

[in] coordinata Y dell'angolo superiore sinistro che forma il rettangolo.

x2

[in] coordinata X dell'angolo in basso a destra che forma il rettangolo.

y2

[in] coordinata Y dell'angolo in basso a destra che forma il rettangolo.

x3

[in] coordinata X del primo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

y3

[in] coordinata Y del primo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

x4

[in] coordinata X del secondo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

y4

[in] coordinata Y del secondo punto, al quale viene disegnata una linea dal centro rettangolare per ottenere il limite dell'arco.

clr

[in] Colore del bordo del settore nel formato ARGB.

fill_clr

[in] Colore di riempimento del settore nel formato ARGB. Usa la funzione ColorToARGB() per convertire un colore nel formato ARGB.

Disegna un settore riempito di un'ellisse con centro a punto (x, y), incisi nel rettangolo, con raggi x e ry. I confini del settore vengono tagliati dal centro dell'ellisse da raggi formati da angoli fi3 e fi4.

```
void  Pie(
   int          x,        // coordinate X del centro ellisse
   int          y,        // coordinate Y del centro ellisse
   int          rx,       // raggio di ellisse sull'asse X
   int          ry,       // raggio di ellisse sull'asse Y
   int          fi3,      // angolo di raggio dal centro di ellisse, che definisce il pr
   int          fi4,      // angolo di raggio dal centro di ellisse, che definisce il se
   const uint   clr,      // colore linea
   const uint   fill_clr // colore riempimento
   );
```

Disegna un settore riempito di un'ellisse con centro al punto (x, y), inciso in rettangolo, con raggi rx e ry e restituisce anche le coordinate dei limiti dell'arco. I confini del settore vengono tagliati dal centro dell'ellisse da raggi formati da angoli fi3 e fi4.

```
void  Pie(
   int          x,        // coordinate X del centro ellisse
   int          y,        // coordinate Y del centro ellisse
   int          rx,       // raggio di ellisse sull'asse X
   int          ry,       // raggio di ellisse sull'asse Y
   int          fi3,      // angolo di raggio dal centro di ellisse, che definisce il pr
   int          fi4,      // angolo di raggio dal centro di ellisse, che definisce il se
   int&         x3,       // coordinate X del primo contorno d'arco
   int&         y3,       // coordinate Y del primo contorno d'arco
   int&         x4,       // coordinate X del secondo contorno d'arco
   int&         y4,       // coordinate Y del secondo contorno d'arco
   const uint   clr,      // colore linea
   const uint   fill_clr // colore riempimento
   );
```

Parametri

x

[in] coordinate X del centro ellisse.

y

[in] coordinate Y del centro ellisse.

rx

[in] Raggio dell'ellisse sull'asse Y, in pixel.

ry

[in] Raggio dell'ellisse sull'asse Y, in pixel.

fi3

[in] Angolo in radianti, che definisce il primo limite dell'arco.

fi4

[in] Angolo in radianti, che definisce il secondo limite dell'arco.

x3

[out] Variabile per ottenere la coordinata X del primo limite dell'arco.

y3

[out] Variabile per ottenere la coordinata Y del primo limite dell'arco.

x4

[out] Variabile per ottenere la coordinata X del secondo limite dell'arco.

y4

[out] Variabile per ottenere la coordinata Y del secondo limite dell'arco.

clr

[in] Colore del bordo del settore nel formato ARGB.

fill_clr

[in] Colore di riempimento del settore nel formato ARGB. Usa la funzione ColorToARGB() per convertire un colore nel formato ARGB.

Esempi di chiamata dei metodi della classe:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Funzione Script Programma Script                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- crea canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- pulisce canvas
   canvas.Erase(clrWhite);
//--- disegna rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- disegna la prima torta
   canvas.Pie(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrBlue),ColorToARGB(c
//--- disegna la seconda torta
   canvas.Pie(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,ColorToARGB(clrGreen),Colo
//--- mostra canvas aggiornato
   canvas.Update();
   DebugBreak();
  }
```

FillPolygon

Disegna un poligono pieno.

```
void  FillPolygon(
   int&          x,      // array con le coordinate X di punti poligonali
   int&          y,      // array con le coordinate Y di punti poligonali
   const uint    clr     // colore
   );
```

Parametri

x

[in] Array delle coordinate X dei punti poligonali.

y

[in] Array delle coordinate Y dei punti poligonali.

clr

[In] Colore in formato ARGB.

FillEllipse

Draws a filled ellipse inscribed in a rectangle with the specified coordinates.

```
void  FillPolygon(
   int          x1,      // coordinate X dell' angolo superiore sinistro del rettangolo
   int          y1,      // coordinate Y dell' angolo superiore sinistro del rettangolo
   int          x2,      // coordinate X dell' angolo inferiore destro del rettangolo
   int          y2,      // coordinate Y dell' angolo superiore sinistro del rettangolo
   const uint   clr      // colore ellisse
   );
```

Parametri

x1

[in] coordinata X dell'angolo superiore sinistro che forma il rettangolo.

y1

[in] coordinata Y dell'angolo superiore sinistro che forma il rettangolo.

x2

[in] coordinata X dell'angolo in basso a destra che forma il rettangolo.

y2

[in] coordinata Y dell'angolo in basso a destra che forma il rettangolo.

clr

[In] Colore in formato ARGB.

GetDefaultColor

Restituisce un colore predefinito per il suo indice.

```
static uint  GetDefaultColor(
   const uint  i      // indice
   );
```

Parametri

i

[in] Indice per ottenere il colore.

Valore di ritorno

Colore.

ChartObjectName

Riceve il nome di un oggetto grafico associato.

```
string  ChartObjectName();
```

Valore di ritorno

il nome di un oggetto grafico legato

Circle

Disegna un cerchio

```
void  Circle(
   int          x,        // X coordinate
   int          y,        // Y coordinate
   int          r,        // raggio
   const uint   clr       // colore
   );
```

Parametri

x

[in] Coordinata X del centro del cerchio.

y

[in] Coordinata Y del centro del cerchio.

r

[in] Raggio del cerchio.

clr

[in] Colore in formato ARGB.

CircleAA

Disegna un cerchio utilizzando l'algoritmo di antialiasing

```
void  CircleAA(
   const int      x,        // X coordinate
   const int      y,        // Y coordinate
   const double   r,        // reggio
   const uint     clr       // colore
   );
```

Parametri

x

[in] Coordinata X del centro del cerchio.

y

[in] Coordinata Y del centro del cerchio.

r

[in] Raggio del cerchio.

clr

[in] Colore in formato ARGB.

CircleWu

Disegna un cerchio utilizzando l'algoritmo di anti-aliasing di Wu

```
void  CircleWu(
   const int      x,        // X coordinate
   const int      y,        // Y coordinate
   const double   r,        // reggio
   const uint     clr       // colore
   );
```

Parametri

x

[in] Coordinata X del centro del cerchio.

y

[in] Coordinata Y del centro del cerchio.

r

[in] Raggio del cerchio.

clr

[in] Colore in formato ARGB.

Create

Crea una risorsa grafica senza legarsi a un oggetto grafico.

```
virtual bool  Create(
   const string       name,                              // nome
   const int          width,                             // larghezza
   const int          height,                            // altezza
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato
   );
```

Parametri

name

[in] Base per un nome di risorsa grafica. Un nome di risorsa viene generato durante la creazione con l'aggiunta di una stringa pseudocasuale.

width

[in] Larghezza (dimensione lungo l'asse X) in pixel.

height

[in] Altezza (dimensione lungo l'asse Y) in pixel.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[In] Metodo di elaborazione colori. Vedere la descrizione della funzione ResourceCreate() per saperne di più sui metodi di elaborazione dei colori.

Valore di ritorno

true - successo, altrimenti - false

CreateBitmap

Crea una risorsa grafica legata a un oggetto chart.

1. Crea una risorsa grafica nella finestra principale del chart corrente.

```
bool  CreateBitmap(
   const string       name,                                  // nome
   const datetime     time,                                  // orario
   const double       price,                                 // prezzo
   const int          width,                                 // larghezza
   const int          height,                                // altezza
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // formato
   );
```

2. Crea una risorsa grafica utilizzando un ID del chart ed un numero di sottofinestra.

```
bool  CreateBitmap(
   const long         chart_id,                              // chart ID
   const int          subwin,                                // numero sottofinestra
   const string       name,                                  // nome
   const datetime     time,                                  // orario
   const double       price,                                 // prezzo
   const int          width,                                 // larghezza
   const int          height,                                // altezza
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // formato
   );
```

Parametri

chart_id

[in] ID del Chart per creare un oggetto .

subwin

[in] Numero sottofinestra del chart per la creazione di un oggetto.

name

[in] Nome dell'oggetto Chart e una base per un nome di risorsa grafica.

time

[in] Coordinate temporali dell'oggetto del chart.

price

[in] Coordinate prezzo dell'oggetto del chart.

width

[in] Larghezza della risorsa grafica (dimensione lungo l'asse X) in pixel.

height

[in] Altezza della risorsa grafica (dimensione lungo l'asse Y) in pixel.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[In] Metodo di elaborazione colori. Vedere la descrizione della funzione ResourceCreate() per saperne di più sui metodi di elaborazione dei colori.

Valore di ritorno

true - successo, altrimenti - false

Nota

Se si utilizza la prima versione funzione, l'oggetto viene creato nella finestra principale del chart corrente.

Le dimensioni dell'oggetto coincidono con la dimensione di una risorsa grafica.

CreateBitmapLabel

Crea una risorsa grafica legata a un oggetto chart.

1. Crea una risorsa grafica nella finestra principale del chart corrente.

```
bool  CreateBitmapLabel(
   const string       name,                              // nome
   const int          x,                                 // X coordinate
   const int          y,                                 // Y coordinate
   const int          width,                             // larghezza
   const int          height,                            // altezza
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato
   );
```

2. Crea una risorsa grafica utilizzando un ID del chart ed un numero di sottofinestra.

```
bool  CreateBitmapLabel(
   const long         chart_id,                          // chart ID
   const int          subwin,                            // numero sottofinestra
   const string       name,                              // nome
   const int          x,                                 // X coordinate
   const int          y,                                 // Y coordinate
   const int          width,                             // larghezza
   const int          height,                            // altezza
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato
   );
```

Parametri

chart_id

[in] ID del Chart per creare un oggetto .

subwin

[in] Numero sottofinestra del chart per la creazione di un oggetto.

name

[in] Nome dell'oggetto Chart e una base per un nome di risorsa grafica.

x

[in] Punto di ancoraggio delle coordinate X dell oggetto Chart.

y

[in] Punto di ancoraggio delle coordinate Y dell oggetto Chart.

width

[in] Larghezza della risorsa grafica (dimensione lungo l'asse X) in pixel.

height

[in] Altezza della risorsa grafica (dimensione lungo l'asse Y) in pixel.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[In] Metodo di elaborazione colori. Vedere la descrizione della funzione ResourceCreate() per saperne di più sui metodi di elaborazione dei colori.

Valore di ritorno

true - successo, altrimenti - false

Nota

Se si utilizza la prima versione funzione, l'oggetto viene creato nella finestra principale del chart corrente.

Le dimensioni dell'oggetto coincidono con la dimensione di una risorsa grafica.

Destroy

Distrugge una risorsa grafica.

```
void  Destroy();
```

Nota

Se una risorsa grafica è stata legata ad un oggetto chart, quest'ultimo viene eliminato.

Ellisse

Disegna un'ellisse utilizzando due punti.

```
void  Ellipse(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinate X del primo punto formando una ellisse.

y1

[in] Coordinate Y del primo punto formando una ellisse.

x2

[in] Coordinate X del secondo punto formando una ellisse.

y2

[in] Coordinate Y del secondo punto formando una ellisse.

clr

[in] Colore in formato ARGB.

EllipseAA

Disegna un'ellisse in base a due punti utilizzando l'algoritmo di anti-aliasing.

```
void  EllipseAA(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinate X del primo punto formando una ellisse.

y1

[in] Coordinate Y del primo punto formando una ellisse.

x2

[in] Coordinate X del secondo punto formando una ellisse.

y2

[in] Coordinate Y del secondo punto formando una ellisse.

clr

[in] Colore in formato ARGB.

EllipseWu

Disegna un'ellisse in base a due punti utilizzando l'algoritmo di anti-aliasing di Wu.

```
void  EllipseWu(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinate X del primo punto formando una ellisse.

y1

[in] Coordinate Y del primo punto formando una ellisse.

x2

[in] Coordinate X del secondo punto formando una ellisse.

y2

[in] Coordinate Y del secondo punto formando una ellisse.

clr

[in] Colore in formato ARGB.

Elimina

Cancella o riempie con il colore specificato.

```
void  Erase(
   const uint  clr=0      // colore
   );
```

Parametri

clr=0

[in] Colore in formato ARGB.

Fill

Riempie un'area.

```
void  Fill(
   int          x,        // X coordinate
   int          y,        // Y coordinate
   const uint   clr       // colore
   );
```

Parametri

x

[in] coordinata X di riempimento punto di partenza.

y

[in] coordinata Y di riempimento punto di partenza.

clr

[in] Colore in formato ARGB.

FillCircle

Disegna un cerchio pieno.

```
void  FillCircle(
   int          x,        // X coordinate
   int          y,        // Y coordinate
   int          r,        // raggio
   const uint   clr       // colore
   );
```

Parametri

x

[in] coordinata X di un centro di cerchio riempito.

y

[in] coordinata Y di un centro di cerchio riempito.

r

[in] raggio del cerchio riempito.

clr

[in] Colore in formato ARGB.

FillRectangle

Disegna un rettangolo riempito.

```
void  FillRectangle(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinate X del primo punto che forma un rettangolo.

y1

[in] Coordinate Y del primo punto che forma un rettangolo.

x2

[in] Coordinate X del secondo punto che forma un rettangolo.

y2

[in] Coordinate Y del secondo punto che forma un rettangolo.

clr

[in] Colore in formato ARGB.

FillTriangle

Disegna un triangolo pieno.

```
void  FillTriangle(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   int          x3,      // X coordinate
   int          y3,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinata X del primo angolo del triangolo.

y1

[in] Coordinata Y del primo angolo del triangolo.

x2

[in] Coordinata X del secondo angolo del triangolo.

y2

[in] Coordinata Y del secondo angolo del triangolo.

x3

[in] Coordinata X del terzo angolo del triangolo.

y3

[in] Coordinata Y del terzo angolo del triangolo.

clr

[in] Colore in formato ARGB.

FontAngleGet

Riceve angolo di inclinazione dei caratteri.

```
uint  FontAngleGet();
```

Valore di ritorno

angolo di inclinazione del carattere

FontAngleSet

Imposta l'angolo di inclinazione dei caratteri.

```
bool  FontAngleSet(
   uint  angle      // angolo
   );
```

Parametri

angle

[in] Angolo di inclinazione del carattere, espresso in decimi di grado.

Valore di ritorno

true - successo, altrimenti - false

FontFlagsGet

Riceve flag dei font.

```
uint  FontFlagsGet();
```

Valore di ritorno

font flags

FontFlagsSet

Imposta flags font.

```
bool  FontFlagsSet(
   uint  flags      // flags
   );
```

Parametri

flags

[in] Flags creazione Font. Vedere la descrizione della funzione TextSetFont() per imparare di più sui flags.

Valore di ritorno

true - successo, altrimenti - false

FontGet

Riceve i parametri del carattere corrente.

```
void  FontGet(
   string&  name,       // nome
   int&     size,       // grandezza
   uint&    flags,      // flags
   uint&    angle       // angolo d'inclinazione
   );
```

Parametri

name

[out] Riferimento alla variabile per la restituzione del nome del carattere.

size

[out] Riferimento alla variabile per la restituzione della grandezza del carattere.

flags

[out] Riferimento alla variabile per la restituzione del flag del carattere.

angle

[out] Riferimento alla variabile per la restituzione dell'angolo di inclinazione del carattere.

FontNameGet

Riceve nome del font.

```
string  FontNameGet();
```

Valore di ritorno

font name

FontNameSet

Imposta nome del font.

```
bool  FontNameSet(
   string  name      // nome
   );
```

Parametri

name

[in] Nome del font. Ad esempio, "Arial".

Valore di ritorno

true - successo, altrimenti - false

FontSet

Imposta il font corrente.

```
bool  FontSet(
   const string  name,         // nome
   const int     size,         // grandezza
   const uint    flags=0,      // flags
   const uint    angle=0       // angolo
   );
```

Parametri

name

[in] Nome del font. Ad esempio, "Arial".

size

[in] Grandezza Font. Vedere la descrizione della funzione TextSetFont() per saperne di più su come impostare una grandezza.

flags=0

[in] Flags creazione Font. Vedere la descrizione della funzione TextSetFont() per imparare di più sui flags.

angle=0

[in] Angolo di inclinazione del carattere, espresso in decimi di grado.

Valore di ritorno

true - successo, altrimenti - false

FontSizeGet

Riceve la grandezza del carattere.

```
int  FontSizeGet();
```

Valore di ritorno

font size

FontSizeSet

Imposta la grandezza del carattere.

```
bool  FontSizeSet(
   int  size      // grandezza
   );
```

Parametri

size

[in] Grandezza Font. Vedere la descrizione della funzione TextSetFont() per saperne di più su come impostare una grandezza.

Valore di ritorno

true - successo, altrimenti - false

Altezza

Riceve l'altezza di una risorsa grafica.

```
int  Height();
```

Valore di ritorno

altezza di una risorsa grafica

Line

Disegna un segmento di una linea a mano libera.

```
void  Line(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinata X del primo punto del segmento.

y1

[in] Coordinata Y del primo punto del segmento.

x2

[in] Coordinata X del secondo punto del segmento.

y2

[in] Coordinata Y del secondo punto del segmento.

clr

[in] Colore in formato ARGB.

LineAA

Disegna un segmento di una linea a mano libera utilizzando l'algoritmo di antialiasing.

```
void  LineAA(
   const int   x1,                  // X coordinate
   const int   y1,                  // Y coordinate
   const int   x2,                  // X coordinate
   const int   y2,                  // Y coordinate
   const uint  clr,                 // colore
   const uint  style=UINT_MAX       // stile linea
   );
```

Parametri

x1

[in] Coordinata X del primo punto del segmento.

y1

[in] Coordinata Y del primo punto del segmento.

x2

[in] Coordinata X del secondo punto del segmento.

y2

[in] Coordinata Y del secondo punto del segmento.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

LineWu

Disegna un segmento di una linea a mano libera utilizzando l'algoritmo di anti-aliasing di Wu.

```
void  LineWu(
   const int   x1,                  // X coordinate
   const int   y1,                  // Y coordinate
   const int   x2,                  // X coordinate
   const int   y2,                  // Y coordinate
   const uint  clr,                 // colore
   const uint  style=UINT_MAX       // stile linea
   );
```

Parametri

x1

[in] Coordinata X del primo punto del segmento.

y1

[in] Coordinata Y del primo punto del segmento.

x2

[in] Coordinata X del secondo punto del segmento.

y2

[in] Coordinata Y del secondo punto del segmento.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

LineHorizontal

Disegna un segmento di una linea orizzontale.

```
void  LineHorizontal(
   int          x1,       // X coordinate
   int          x2,       // X coordinate
   int          y,        // Y coordinate
   const uint   clr       // colore
   );
```

Parametri

x1

[in] Coordinata X del primo punto del segmento.

x2

[in] Coordinata X del secondo punto del segmento.

y

[in] Coordinata Y del segmento.

clr

[in] Colore in formato ARGB.

LineVertical

Disegna un segmento di una linea verticale.

```
void  LineVertical(
   int          x,        // X coordinate
   int          y1,       // Y coordinate
   int          y2,       // Y coordinate
   const uint   clr       // colore
   );
```

Parametri

x

[in] Coordinata X del segmento.

y1

[in] Coordinata Y del primo punto del segmento.

y2

[in] Coordinata Y del secondo punto del segmento.

clr

[in] Colore in formato ARGB.

LineStyleSet

Imposta lo stile della linea.

```
void  LineStyleSet(
   const uint  style      // stile
   );
```

Parametri

style

[in] Stile della linea.

Nota

Il parametro di input può avere qualsiasi valore dell' enumerazione ENUM_LINE_STYLE. Inoltre, è possibile creare uno stile di disegno linea personalizzato.

LineThick

Disegna un segmento di una linea a mano libera con una larghezza specificata utilizzando l'algoritmo di antialiasing.

```
void  LineThick(
   const int      x1,            // coordinate X del primo punto del segmento
   const int      y1,            // coordinate Y del primo punto del segmento
   const int      x2,            // coordinate X del secondo punto del segmento
   const int      y2,            // coordinate Y del secondo punto del segmento
   const uint     clr,           // colore
   const int      size,          // spessore linea
   const uint     style,         // stile linea
   ENUM_LINE_END  end_style      // stile fine linea
   )
```

Parametri

x1

[in] coordinate X del primo punto del segmento.

y1

[in] coordinate Y del primo punto del segmento.

x2

[in] coordinate X del secondo punto del segmento.

y2

[in] coordinate Y del secondo punto del segmento.

clr

[in] Colori in formato ARGB.

size

[in] Larghezza linea.

style

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_STYLE o un valore personalizzato.

end_style

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_END

ENUM_LINE_END

ID	Descrizione
LINE_END_ROUND	Le estremità della linea sono arrotondate.
LINE_END_BUTT	Le estremità della linea vengono tagliate.

ID	Descrizione
LINE_END_SQUARE	Una linea termina in un rettangolo pieno.

LineThickVertical

Disegna un segmento verticale di una linea libera che ha una larghezza specificata usando algoritmo antialiasing.

```
void  LineThickVertical(
   const int      x,             // coordinate X del segmento
   const int      y1,            // coordinate Y del primo punto del segmento
   const int      y2,            // coordinate Y del secondo punto del segmento
   const uint     clr,           // colore
   const int      size,          // spessore linea
   const uint     style,         // stile linea
   ENUM_LINE_END  end_style      // stile fine linea
   )
```

Parametri

x

[in] Coordinate X del segmento.

y1

[in] coordinate Y del primo punto del segmento.

y2

[in] coordinate Y del secondo punto del segmento.

clr

[in] Colori in formato ARGB.

size

[in] Larghezza linea.

style

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_STYLE o un valore personalizzato.

end_style

[in] Lo stile della linea è uno dei valori dell'enumeratore ENUM_LINE_END.

LineThickHorizontal

Disegna un segmento orizzontale di una linea libera che ha un antialiasing di larghezza specificato.

```
void  LineThickHorizontal(
   const int      x1,            // coordinate X coordinate del primo punto del segmer
   const int      x2,            // coordinate X del secondo punto del segmento
   const int      y,             // coordinate Y coordinate del segmento
   const uint     clr,           // colore
   const int      size,          // spessore linea
   const uint     style,         // stile linea
   ENUM_LINE_END  end_style      // stile fine linea
   )
```

Parametri

x1

[in] coordinate X del primo punto del segmento.

x2

[in] coordinate X del secondo punto del segmento.

y

[in] Coordinate Y del segmento.

clr

[in] Colori in formato ARGB.

size

[in] Larghezza linea.

style

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_STYLE o un valore personalizzato.

end_style

[in] Lo stile della linea è uno dei valori dell'enumeratore ENUM_LINE_END.

LoadFromFile

Legge un'immagine da un file BMP.

```
bool  LoadFromFile(
   const string  filename      // nome del file
   );
```

Parametri

filename

[in] Nome del File (inclusa estensione "BMP").

Valore di ritorno

true - successo, altrimenti - false

PixelGet

Riceve il colore del punto con le coordinate specificate.

```
uint  PixelGet(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   );
```

Parametri

x

[in] Coordinata X del punto.

y

[in] Coordinata Y del punto.

Valore di ritorno

Colore del Punto in formato ARGB.

PixelSet

Imposta il colore del punto con le coordinate specificate.

```
void  PixelSet(
   const int   x,        // X coordinate
   const int   y,        // Y coordinate
   const uint  clr       // colore
   );
```

Parametri

x

[in] Coordinata X del punto.

y

[in] Coordinata Y del punto.

clr

[in] Colore in formato ARGB.

PixelSetAA

Disegna un punto utilizzando l'algoritmo di antialiasing.

```
void  PixelSetAA(
   const double  x,        // X coordinate
   const double  y,        // Y coordinate
   const uint    clr       // colore
   );
```

Parametri

x

[in] Coordinata X del punto.

y

[in] Coordinata Y del punto.

clr

[in] Colore in formato ARGB.

Polygon

Disegna un poligono.

```
void  Polygon(
   int&         x[],     // array delle coordinate X
   int&         y[],     // array delle coordinate Y
   const uint   clr      // colore
   );
```

Parametri

x[]

[in] Array delle coordinate X di un punto del poligono.

y[]

[in] Array delle coordinate Y di un punto del poligono.

clr

[in] Colore in formato ARGB.

PolygonAA

Disegna un poligono utilizzando l'algoritmo di antialiasing.

```
void  PolygonAA(
   int&         x[],                // array delle coordinate X
   int&         y[],                // array delle coordinate Y
   const uint   clr,                // colore
   const uint   style=UINT_MAX      // stile linea
   );
```

Parametri

x[]

[in] Array delle coordinate X di un punto del poligono.

y[]

[in] Array delle coordinate Y di un punto del poligono.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

PolygonWu

Disegna un poligono utilizzando l'algoritmo di anti-aliasing di Wu.

```
void  PolygonWu(
   int&         x[],                // array delle coordinate X
   int&         y[],                // array delle coordinate Y
   const uint   clr,                // colore
   const uint   style=UINT_MAX      // stile linea
   );
```

Parametri

x[]

[in] Array delle coordinate X di un punto del poligono.

y[]

[in] Array delle coordinate Y di un punto del poligono.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

PolygonThick

Disegna un poligono con una larghezza specifica utilizzando un algoritmo antialiasing.

```
void  PolygonThick(
   const int&      x[],          // array con le coordinate X dei punti del poligono
   const int&      y[],          // array con le coordinate Y dei punti del poligono
   const uint      clr,          // colore
   const int       size,         // spessore linea
   const uint      style,        // stile linea
   ENUM_LINE_END   end_style     // stile fine linea
   )
```

Parametri

x[]

[in] Array delle coordinate X dei punti del poligono.

y[]

[in] Array delle coordinate Y dei punti del poligono.

clr

[in] Colori in formato ARGB.

size

[in] Larghezza linea.

style

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_STYLE o un valore personalizzato.

end_style

[in] Lo stile della linea è uno dei valori dell'enumeratore ENUM_LINE_END.

PolygonSmooth

Disegna un poligono con una larghezza specificata consecutivamente utilizzando due algoritmi antialiasing. In primo luogo, i singoli segmenti vengono lisciati in base alle curve di Bezier. Quindi, l'algoritmo di antialiasing raster viene applicato al poligono costruito da questi segmenti per migliorare la qualità di rendering.

```
void  PolygonSmooth(
   int&              x[],                        // array con le coordinate X coordi
   int&              y[],                        // array con le coordinate Y dei pu
   const uint        clr,                        // colore
   const int         size,                       // spessore della linea
   ENUM_LINE_STYLE   style=STYLE_SOLID,          // stile della linea
   ENUM_LINE_END     end_style=LINE_END_ROUND,   // stile del fine linea
   double            tension=0.5,                // valori del parametro antialiasin
   double            step=10                     // lunghezza delle linee approssima
   )
```

Parametri

&x[]

[in] Array delle coordinate X dei punti del poligono.

&y[]

[in] Array delle coordinate Y dei punti del poligono.

clr

[in] Colori in formato ARGB.

size

[in] Larghezza linea.

style=STYLE_SOLID

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_STYLE o un valore personalizzato.

end_style=LINE_END_ROUND

[in] Stile della linea è uno dei valori dell'enumerazione ENUM_LINE_END.

tension=0.5

[in] Valori del parametro smussamento.

step=10

[in] Lunghezza delle linee di approssimazione.

Polyline

Disegna una polilinea.

```
void  Polyline(
   int&         x[],     // array delle coordinate X
   int&         y[],     // array delle coordinate Y
   const uint   clr      // colore
   );
```

Parametri

x[]

[in] Array delle coordinate X di una polilinea.

y[]

[in] Array delle coordinate Y di una polilinea.

clr

[in] Colore in formato ARGB.

PolylineSmooth

Disegna una polilinea con una larghezza specifica consecutivamente utilizzando due algoritmi antialiasing. In primo luogo, i singoli segmenti di linea vengono smussati in base alle curve di Bezier. Quindi, l'algoritmo di antialiasing raster viene applicato alla polilinea costruita da questi segmenti per migliorare la qualità di rendering.

```
void  PolylineSmooth(
   const int&        x[],                        // array con le coordinate X dei punt
   const int&        y[],                        // array con le coordinate Y dei punt
   const uint        clr,                        // colore
   const int         size,                       // spessore linea
   ENUM_LINE_STYLE   style=STYLE_SOLID,          // stile linea
   ENUM_LINE_END     end_style=LINE_END_ROUND,   // stile del fine linea
   double            tension=0.5,                // valore parametro antialiasing
   double            step=10                     // step di approssimazione
   )
```

Parametri

&x[]

[in] Array di coordinate X di una polilinea.

&y[]

[in] Array di coordinate Y di una polilinea.

clr

[in] Colori in formato ARGB.

size

[in] Larghezza linea.

style=STYLE_SOLID

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_STYLE o un valore personalizzato.

end_style=LINE_END_ROUND

[in] Lo stile della linea è uno dei valori dell'enumeratore ENUM_LINE_END.

tension=0.5

[in] Valori del parametro smussamento.

step=10

[in] Step d'approssimazione.

PolylineThick

Disegna una polilinea con una larghezza specificata utilizzando un algoritmo antialiasing.

```
void  PolylineThick(
   const int       &x[],          // array con le coordinate coordinate X dei punti del
   const int       &y[],          // array con le coordinate Y dei punti del polilinea
   const uint      clr,           // colore
   const int       size,          // spessore linea
   const uint      style,         // stile linea
   ENUM_LINE_END   end_style      // stile fine linea
   )
```

Parametri

&x[]

[in] Array di coordinate X di una polilinea.

&y[]

[in] Array di coordinate Y di una polilinea.

clr

[in] Colori in formato ARGB.

size

[in] Larghezza linea.

style

[in] Lo stile della linea è uno dei valori della enumerazione ENUM_LINE_STYLE o un valore personalizzato.

end_style

[in] Lo stile linea è uno dei valori dell'enumerazione ENUM_LINE_END

PolylineWu

Disegna una polilinea utilizzando l'algoritmo di anti-aliasing di Wu.

```
void  PolylineWu(
   int&         x[],                // array delle coordinate X
   int&         y[],                // array delle coordinate Y
   const uint   clr,                // colore
   const uint   style=UINT_MAX      // stile linea
   );
```

Parametri

x[]

[in] Array delle coordinate X di una polilinea.

y[]

[in] Array delle coordinate Y di una polilinea.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

PolylineAA

Disegna una polilinea utilizzando l'algoritmo di antialiasing.

```
void  PolylineAA(
   int&         x[],                // array delle coordinate X
   int&         y[],                // array delle coordinate Y
   const uint   clr,                // colore
   const uint   style=UINT_MAX      // stile linea
   );
```

Parametri

x[]

[in] Array delle coordinate X di una polilinea.

y[]

[in] Array delle coordinate Y di una polilinea.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

Rectangle

Disegna un rettangolo con due punti.

```
void  Rectangle(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinate X del primo punto che forma un rettangolo.

y1

[in] Coordinate Y del primo punto che forma un rettangolo.

x2

[in] Coordinate X del secondo punto che forma un rettangolo.

y2

[in] Coordinate Y del secondo punto che forma un rettangolo.

clr

[in] Colore in formato ARGB.

Ridimensiona

Ridimensiona una risorsa grafica.

```
bool  Resize(
   const int  width,       // larghezza
   const int  height       // altezza
   );
```

Parametri

width

[in] Nuova larghezza di una risorsa grafica.

height

[in] Nuova altezza di una risorsa grafica.

Valore di ritorno

true - successo, altrimenti - false

Nota

Quando si fa il ridimensionamento, l'immagine precedente non viene salvata.

ResourceName

Riceve il nome di una risorsa grafica.

```
string  ResourceName();
```

Valore di ritorno

nome di una risorsa grafica

TextHeight

Riceve l'altezza del testo.

```
int  TextHeight(
   const string  text      // testo
   );
```

Parametri

text

[in] Testo per la misurazione.

Valore di ritorno

altezza del testo in pixel

Nota

Il tipo di carattere corrente viene utilizzato per misurare il testo.

TextOut

Visualizza il testo.

```
void  TextOut(
   int          x,               // X coordinate
   int          y,               // Y coordinate
   string       text,            // testo
   const uint   clr,             // colore
   uint         alignment=0      // allineamento
   );
```

Parametri

x

[in] Coordinata X di ancoraggio di testo.

y

[in] Coordinata Y di ancoraggio di testo.

text

[in] Testo da visualizzare.

clr

[in] Colore in formato ARGB.

alignment=0

[in] Metodo di ancoraggio del testo. Vedere la descrizione della funzione TextOut() per saperne di più sui metodi di ancoraggio.

Nota

Il tipo di carattere corrente viene utilizzato per visualizzare il testo.

TextSize

Riceve la dimensione del testo.

```
void  TextSize(
   const string  text,        // testo
   int&          width,       // larghezza
   int&          height       // altezza
   );
```

Parametri

text

[in] Testo per la misurazione.

width

[out] Riferimento alla variabile per la restituzione della larghezza del testo.

height

[out] Riferimento alla variabile per la restituzione dell'altezza del testo.

Nota

Il carattere corrente viene utilizzato per misurare il testo.

TextWidth

Riceve la larghezza del testo.

```
int  TextWidth(
   const string  text      // testo
   );
```

Parametri

text

[in] Testo per la misurazione.

Valore di ritorno

altezza del testo in pixel

Nota

Il carattere corrente viene utilizzato per misurare il testo.

TransparentLevelSet

Imposta il livello di trasparenza.

```
void  TransparentLevelSet(
   const uchar  value      // valore
   );
```

Parametri

value

[in] Nuovo valore del livello di trasparenza.

Nota

0 sta per la massima trasparenza, mentre 255 - per la piena opacità.

L'impostazione di un livello di trasparenza riguarda tutto ciò che è stato precedentemente disegnato. Il livello di trasparenza non influisce ulteriori costruzioni.

Triangle

Disegna un triangolo.

```
void  Triangolo(
   int          x1,      // X coordinate
   int          y1,      // Y coordinate
   int          x2,      // X coordinate
   int          y2,      // Y coordinate
   int          x3,      // X coordinate
   int          y3,      // Y coordinate
   const uint   clr      // colore
   );
```

Parametri

x1

[in] Coordinata X del primo angolo del triangolo.

y1

[in] Coordinata Y del primo angolo del triangolo.

x2

[in] Coordinata X del secondo angolo del triangolo.

y2

[in] Coordinata Y del secondo angolo del triangolo.

x3

[in] Coordinata X del terzo angolo del triangolo.

y3

[in] Coordinata Y del terzo angolo del triangolo.

clr

[in] Colore in formato ARGB.

TriangleAA

Disegna un triangolo utilizzando l'algoritmo di antialiasing.

```
void  TriangleAA(
   const int   x1,                  // X coordinate
   const int   y1,                  // Y coordinate
   const int   x2,                  // X coordinate
   const int   y2,                  // Y coordinate
   const int   x3,                  // X coordinate
   const int   y3,                  // Y coordinate
   const uint  clr,                 // colore
   const uint  style=UINT_MAX       // stile linea
   );
```

Parametri

x1

[in] Coordinata X del primo angolo del triangolo.

y1

[in] Coordinata Y del primo angolo del triangolo.

x2

[in] Coordinata X del secondo angolo del triangolo.

y2

[in] Coordinata Y del secondo angolo del triangolo.

x3

[in] Coordinata X del terzo angolo del triangolo.

y3

[in] Coordinata Y del terzo angolo del triangolo.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

TriangleWu

Disegna un triangolo utilizzando l'algoritmo di anti-aliasing di Wu.

```
void  TriangleWu(
   const int   x1,                  // X coordinate
   const int   y1,                  // Y coordinate
   const int   x2,                  // X coordinate
   const int   y2,                  // Y coordinate
   const int   x3,                  // X coordinate
   const int   y3,                  // Y coordinate
   const uint  clr,                 // colore
   const uint  style=UINT_MAX       // stile linea
   );
```

Parametri

x1

[in] Coordinata X del primo angolo del triangolo.

y1

[in] Coordinata Y del primo angolo del triangolo.

x2

[in] Coordinata X del secondo angolo del triangolo.

y2

[in] Coordinata Y del secondo angolo del triangolo.

x3

[in] Coordinata X del terzo angolo del triangolo.

y3

[in] Coordinata Y del terzo angolo del triangolo.

clr

[in] Colore in formato ARGB.

style=UINT_MAX

[in] Lo stile della Linea è uno dei valori dell' enumerazione ENUM_LINE_STYLE o un valore personalizzato.

Aggiorna

Consente di visualizzare le modifiche sullo schermo.

```
void  Update(
   const bool  redraw=true      // flag
   );
```

Parametri

redraw=true

Flag per la necessità di ridisegno del chart.

Larghezza

Riceve la larghezza di una risorsa grafica.

```
int  Width();
```

Valore di ritorno

larghezza risorsa grafica

CChartCanvas

Classe base per l'implementazione delle classi, che vengono utilizzate per il disegno di grafici e dei loro elementi.

Descrizione

Questa classe include i metodi per lavorare con gli elementi di base di qualsiasi grafico: assi coordinati e loro marchi, leggenda del grafico, griglia, sfondo, ecc. Qui è possibile personalizzare le opzioni per visualizzare gli elementi: visibilità, colore del testo, ecc.

Dichiarazione

```
class CChartCanvas : public CCanvas
```

Titolo

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

Gerarchia ereditaria

CCanvas

CChartCanvas

Discendenti diretti

CHistogramChart, CLineChart, CPieChart

Metodi di classe

Metodo	Azione
ColorBackground	Restituisce e imposta il colore di sfondo.
ColorBorder	Restituisce e imposta il colore del bordo.
ColorText	Restituisce e imposta il colore del testo.
ColorGrid	Restituisce e imposta il colore della griglia.
MaxData	Restituisce e imposta la quantità massima di dati (serie) consentita.
MaxDescrLen	Restituisce e imposta la lunghezza massima dei descrittori.
ShowFlags	Restituisce e imposta la barra di visibilità degli elementi del grafico.
IsShowLegend	Restituisce e imposta la bandiera di visibilità della leggenda sul grafico.

Metodo	Azione
IsShowScaleLeft	Restituisce la barra di visibilità della scala dei valori a sinistra.
IsShowScaleRight	Restituisce la barra di visibilità della scala dei valori a destra.
IsShowScaleTop	Restituisce la barra di visibilità della scala dei valori in alto.
IsShowScaleBottom	Restituisce la barra di visibilità della scala dei valori nella parte inferiore.
IsShowGrid	Restituisce la barra di visibilità della griglia sul grafico.
IsShowDescriptors	Restituisce la barra di visibilità dei descrittori sul grafico.
IsShowPercent	Restituisce la barra di visibilità delle percentuali sul grafico.
VScaleMin	Restituisce e imposta il minimo sulla scala verticale dei valori.
VScaleMax	Restituisce e imposta il massimo sulla scala verticale dei valori.
NumGrid	Restituisce e imposta il numero di divisioni di scala verticale durante la tracciatura della griglia del grafico.
DataOffset	Restituisce e imposta il valore di offset dei dati.
DataTotal	Restituisce il numero totale di serie di dati nel grafico.
DrawDescriptors	Metodo virtuale per il disegno dei descrittori.
DrawData	Metodo virtuale per la creazione di serie di dati nell'indice specificato.
Create	Metodo virtuale che crea una risorsa grafica.
AllowedShowFlags	Imposta l'insieme di flag di visibilità consentito per gli elementi del grafico.
ShowLegend	Imposta la barra di visibilità per la leggenda.
ShowScaleLeft	Imposta la barra di visibilità per la scala sinistra.
ShowScaleRight	Imposta la barra di visibilità per la giusta scala.

Metodo	Azione
ShowScaleTop	Imposta la barra di visibilità per la scala superiore.
ShowScaleBottom	Imposta la barra di visibilità per la scala inferiore.
ShowGrid	Imposta la barra di visibilità per la griglia.
ShowDescriptors	Imposta la barra di visibilità dei descrittori.
ShowValue	Imposta la barra di visibilità dei valori.
ShowPercent	Imposta la barra di visibilità per le percentuali.
LegendAlignment	Imposta l'allineamento del testo per la leggenda.
Accumulative	Imposta la bolla di accumulo di valore per la serie.
VScaleParams	Imposta i parametri per la scala verticale dei valori.
DescriptorUpdate	Aggiorna il valore del descrittore di serie (nella posizione specificata).
ColorUpdate	Aggiorna i colori della serie (nella posizione specificata).
ValuesCheck	Esegue calcoli interni per la compilazione del grafico.
Redraw	Ridisegna il grafico.
DrawBackground	Disegna lo sfondo.
DrawLegend	Redigera la leggenda.
DrawLegendVertical	Disegna una leggenda verticale.
DrawLegendHorizontal	Disegna una leggenda orizzontale.
CalcScales	Calcola le coordinate della scala.
DrawScales	Redige tutte le scale di valori.
DrawScaleLeft	Riduce la scala sinistra dei valori.
DrawScaleRight	Redige la giusta scala dei valori.
DrawScaleTop	Riduce la scala superiore dei valori
DrawScaleBottom	Riduce la scala del valore inferiore.
DrawGrid	Ridisegna il grafico.
DrawChart	Ridisegna il grafico.

Metodi ereditati da CCanvas

Create, CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

ColorBackground (Metodo Get)

Restituisce il colore di sfondo.

```
uint  ColorBackground()
```

Valore di ritorno

Colore di sfondo.

ColorBackground (Metodo Set)

Imposta il colore di sfondo.

```
void  ColorBackground(
  const uint  value,  // colore sfondo
  )
```

Parametri

valore

[in] Colore sfondo.

ColorBorder (Metodo Get)

Restituisce il colore del bordo.

```
uint  ColorBorder()
```

Valore di ritorno

Colore del bordo.

ColorBorder (Metodo Set)

Imposta il colore del bordo.

```
void  ColorBorder(
  const uint  value,  // colore bordo
  )
```

Parametri

valore

[in] Colore bordo.

ColorText (Metodo Get)

Restituisce il colore del testo.

```
uint  ColorText()
```

Valore di ritorno

Colore del testo.

ColorText (Metodo Set)

Imposta il colore del testo.

```
void  ColorText(
  const uint  value,  // colore del testo
  )
```

Parametri

valore

[in] Testo colore.

ColorGrid (Metodo Get)

Restituisce il colore della griglia.

```
uint  ColorGrid()
```

Valore di ritorno

Colore della griglia.

ColorGrid (Metodo Set)

Imposta il colore della griglia.

```
void  ColorGrid(
  const uint  value,  // colore griglia
  )
```

Parametri

valore

[in] Colore griglia.

MaxData (Metodo Get)

Restituisce la quantità massima di dati (serie) consentita.

```
uint  MaxData()
```

Valore di ritorno

La quantità massima di dati (serie).

MaxData (Metodo Set)

Imposta la quantità massima di dati (serie) consentita.

```
void  MaxData(
  const uint  value,  // ammontare dei dati
  )
```

Parametri

valore

[in] La quantità massima di dati (serie).

MaxDescrLen (Metodo Get)

Restituisce la lunghezza massima dei descrittori.

```
uint  MaxDescrLen()
```

Valore di ritorno

Il valore della lunghezza massima dei descrittori.

MaxDescrLen (Metodo Set)

Imposta la lunghezza massima dei descrittori.

```
void  MaxDescrLen(
  const uint  value,  // lughezza massima
  )
```

Parametri

valore

[in] Il valore di la lunghezza massima dei descrittori.

ShowFlags (Metodo Get)

Restituisce la barra di visibilità degli elementi del grafico.

```
bool  ShowFlags()
```

Valore di ritorno

Valore della barra di visibilità degli elementi del grafico.

ShowFlags (Metodod Set)

Imposta la barra di visibilità degli elementi del grafico.

```
void  ShowFlags(
  const uint  flags,  // flag
  )
```

Parametri

flags

[in] Valore del flag di visibilità degli elementi del grafico.

IsShowLegend

Restituisce e imposta la bandiera di visibilità della leggenda sul grafico.

```
bool  IsShowLegend()
```

Valore di ritorno

true se la legenda è visibile, altrimenti — false.

IsShowScaleLeft

Restituisce la barra di visibilità della scala dei valori a sinistra.

```
bool  IsShowScaleLeft()
```

Valore di ritorno

true se la scala dei valori è visibile, altrimenti — false.

IsShowScaleRight

Restituisce la barra di visibilità della scala dei valori a destra.

```
bool  IsShowScaleRight()
```

Valore di ritorno

true se la scala dei valori è visibile, altrimenti — false.

IsShowScaleTop

Restituisce la barra di visibilità della scala dei valori in alto.

```
bool  IsShowScaleTop()
```

Valore di ritorno

true se la scala dei valori è visibile, altrimenti — false.

IsShowScaleBottom

Restituisce la barra di visibilità della scala dei valori nella parte inferiore.

```
bool  IsShowScaleBottom()
```

Valore di ritorno

true se la scala dei valori è visibile, altrimenti — false.

IsShowGrid

Restituisce la barra di visibilità della griglia sul grafico.

```
bool  IsShowGrid()
```

Valore di ritorno

true se la griglia è visibile, altrimenti — false.

IsShowDescriptors

Restituisce la barra di visibilità dei descrittori sul grafico.

```
bool  IsShowDescriptors()
```

Valore di ritorno

true se i descrittori sono visibili, altrimenti — false.

IsShowPercent

Restituisce la barra di visibilità delle percentuali sul grafico.

```
bool  IsShowPercent()
```

Valore di ritorno

true se le percentuali sono visibili, altrimenti — false.

VScaleMin (Metodo Get)

Restituisce il minimo sulla scala verticale dei valori.

```
double  VScaleMin()
```

Valore di ritorno

Il valore minimo della scala verticale.

VScaleMin (Metodo Set)

Imposta il minimo sulla scala verticale dei valori.

```
void  VScaleMin(
  const double  value,    // valore sulla scala verticale
  )
```

Parametri

valore

[in] Il valore minimo.

VScaleMax

Restituisce il massimo sulla scala verticale dei valori.

```
double  VScaleMax()
```

Valore di ritorno

Il valore massimo sulla scala verticale.

VScaleMax

Imposta il massimo sulla scala verticale dei valori.

```
void  VScaleMax(
  const double  value,  //   valore sulla scala verticale
  )
```

Parametri

valore

[in] Il valore massimo.

NumGrid

Restituisce il numero di divisioni di scala verticale durante la tracciatura della griglia del grafico.

```
uint  NumGrid()
```

Valore di ritorno

Il numero delle divisioni.

NumGrid

Imposta il numero di divisioni di scala verticale durante la tracciatura della griglia del grafico.

```
void  NumGrid(
  const uint  value,  // numero di divisioni
  )
```

Parametri

valore

[in] Il numero di divisioni.

DataOffset

Restituisce il valore di offset dei dati.

```
int  DataOffset()
```

Valore di ritorno

Offset dati.

DataOffset

Imposta il valore di offset dei dati.

```
void  DataOffset(
  const int  value,  // offset
  )
```

Parametri

valore

[in] Offset dati.

DataTotal

Restituisce il numero totale di serie di dati nel grafico.

```
uint  DataTotal()
```

Valore di ritorno

Il numero delle serie.

DrawDescriptors

Metodo virtuale per il disegno dei descrittori.

```
virtual void  DrawDescriptors()
```

DrawData

Metodo virtuale per la creazione di serie di dati nell'indice specificato.

```
virtual void  DrawData(
   const uint  idx=0,  // index
   )
```

Parametri

idx=0

[in] Indice delle serie.

Create

Metodo virtuale che crea una risorsa grafica.

```
virtual bool  Create(
  const string       name,      // nome della risorsa
  const int          width,     // spessore
  const int          height,    // altrzza
  ENUM_COLOR_FORMAT  clrfmt,    // formato
  )
```

Parametri

name

[in] Basi per un nome di risorsa grafica. Un nome di risorsa viene generato durante la creazione aggiungendo una stringa pseudocasuale.

width

[in] Larghezza (dimensione lungo l'asse x) in pixel.

height

[in] Altezza (dimensione lungo l'asse Y) in pixel.

clrfmt

[in] Metodo di elaborazione del colore. Vedere la descrizione della funzione ResourceCreate() per ulteriori informazioni sui metodi di elaborazione dei colori.

Valore di ritorno

true se riuscito, altrimenti — false.

AllowedShowFlags

Imposta l'insieme di flag di visibilità consentito per gli elementi del grafico.

```
void  AllowedShowFlags(
  const uint  flags,  // flags
  )
```

Parametri

flags

[in] Flags consentite.

ShowLegend

Imposta il valore della flag di visibilità per la legenda (FLAG_SHOW_LEGEND).

```
void  ShowLegend(
  const bool  flag,  // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — la legenda diventa visibile.
- false — la legenda diventa invisibile.

ShowScaleLeft

Imposta il valore della barra di visibilità per la scala sinistra (FLAG_SHOW_SCALE_LEFT).

```
void  ShowScaleLeft(
  const bool  flag,  // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — la scala sinistra diventa visibile.
- false — la scala sinistra diventa invisibile.

ShowScaleRight

Imposta il valore della barra di visibilità per la giusta scala (FLAG_SHOW_SCALE_RIGHT).

```
void  ShowScaleRight(
  const bool  flag,  // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — la scala destra diventa visibile.
- false — la scala destra diventa invisibile.

ShowScaleTop

Imposta il valore flag della visibilità per la scala superiore (FLAG_SHOW_SCALE_TOP).

```
void  ShowScaleTop(
  const bool  flag,  // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — la scala superiore diventa visibile.
- false — la scala superiore diventa invisibile.

ShowScaleBottom

Imposta il valore di flag della visibilità per la scala inferiore (FLAG_SHOW_SCALE_BOTTOM).

```
void  ShowScaleBottom(
  const bool  flag,     // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — la scala inferiore diventa visibile.
- false — la scala inferiore diventa invisibile.

ShowGrid

Imposta il valore di flag della visibilità per la griglia (FLAG_SHOW_GRID).

```
void  ShowGrid(
  const bool  flag,  // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — la griglia diventa visibile.
- falso — la griglia diventa invisibile.

ShowDescriptors

Imposta il valore della flag di visibilità per i descrittori (FLAG_SHOW_DESCRIPTORS).

```
void  ShowDescriptors(
  const bool  flag,    // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — il descrittore diventa visibile.
- false — il descrittore diventa invisibile.

ShowValue

Imposta la barra di visibilità dei valori (FLAG_SHOW_VALUE).

```
void  ShowValue(
  const bool  flag,  // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — il valore diventa visibile.
- false — il valore diventa invisibile.

ShowPercent

Imposta il valore della flag di visibilità per le percentuali (FLAG_SHOW_PERCENT).

```
void  ShowPercent(
  const bool  flag,  // valore flag
  )
```

Parametri

flag

[in] Valore flag:

- true — la percentuale diventa visibile.
- false — la percentuale diventa invisibile.

LegendAlignment

Imposta l'allineamento del testo per la leggenda.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // flag
  )
```

Parametri

valore

[in] Ottiene uno dei valori dell'enumerazione ENUM_ALIGNMENT:

- ALIGNMENT_LEFT — allineamento a sinistra.
- ALIGNMENT_TOP — allineamento verso l'alto.
- ALIGNMENT_RIGHT — allineamento a destra.
- ALIGNMENT_BOTTOM — allineamento verso il basso.

Accumulative

Imposta la bolla di accumulo di valore per la serie.

```
void  Accumulative(
  const bool  flag=true,  // valore flag
  )
```

Parametri

flag=true

[in] Valore flag:

- true — il valore corrente della serie viene sostituito dalla somma di tutti i valori precedenti.
- false — la modalità standard per disegnare serie

VScaleParams

Imposta i parametri per la scala verticale dei valori.

```
void  VScaleParams(
  const double  max,   // massimo
  const double  min,   // minimo
  const uint    grid,  // numero di divisioni
  )
```

Parametri

max

[in] Il valore minimo.

min

[in] Il valore massimo.

grid

[in] Il numero di divisioni di scala.

DescriptorUpdate

Aggiorna il valore del descrittore di serie (nella posizione specificata).

```
bool  DescriptorUpdate(
  const uint    pos,    // indice
  const string  descr,  // valore
  )
```

Parametri

pos

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

descr

[in] Valore descrittore.

Valore di ritorno

true se riuscito, altrimenti — false.

ColorUpdate

Aggiorna i colori della serie (nella posizione specificata).

```
bool  ColorUpdate(
  const uint  pos,  // indice
  const uint  clr,  // colore
  )
```

Parametri

pos

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

clr

[in] Valore del color.

Valore di ritorno

true se riuscito, altrimenti — false.

ValuesCheck

Metodo virtuale ausiliario, effettua calcoli interni per la compilazione del grafico.

```
virtual void  ValuesCheck()
```

Redraw

Metodo virtuale per la ridefinizione del grafico.

```
virtual void  Redraw()
```

DrawBackground

Metodo virtuale per la ridefinizione dello sfondo.

```
virtual void  DrawBackground()
```

DrawLegend

Metodo virtuale per la ridefinizione della leggenda.

```
virtual void  DrawLegend()
```

DrawLegendVertical

Disegna una leggenda verticale.

```
int  DrawLegendVertical(
  const int  w,  // spessore
  const int  h,  // altezza
  )
```

Parametri

w

[in] La larghezza massima del testo nella leggenda.

h

[in] L'altezza massima del testo nella leggenda.

Valore di ritorno

Larghezza della leggenda in pixel.

DrawLegendHorizontal

Disegna una leggenda orizzontale.

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
  )
```

Parametri

w

[in] La larghezza massima del testo nella leggenda.

h

[in] L'altezza massima del testo nella leggenda.

Valore di ritorno

Altezza della legenda in pixel.

CalcScales

Metodo virtuale per il calcolo delle coordinate delle etichette per la scala dei valori.

```
virtual void  CalcScales()
```

DrawScales

Metodo virtuale per la ridefinizione di tutte le scale di valori.

```
virtual void  DrawScales()
```

DrawScaleLeft

Metodo virtuale per rieseguire la scala sinistra dei valori.

```
virtual int  DrawScaleLeft(
  const bool  draw,  // flag
  )
```

Parametri

draw

[in] Flag che indica se la scala deve essere ridisegnata.

Valore di ritorno

Larghezza della scala dei valori.

DrawScaleRight

Metodo virtuale per la ridefinizione della scala destra dei valori.

```
virtual int  DrawScaleRight(
  const bool  draw,  // flag
  )
```

Parametri

draw

[in] Flag che indica se la scala deve essere ridisegnata.

Valore di ritorno

Larghezza della scala dei valori.

DrawScaleTop

Metodo virtuale per la ridefinizione della scala superiore dei valori.

```
virtual int  DrawScaleTop(
  const bool  draw,  // flag
  )
```

Parametri

draw

[in] Flag che indica se la scala deve essere ridisegnata.

Valore di ritorno

Altezza della scala dei valori.

DrawScaleBottom

Metodo virtuale per ridisegnare la scala inferiore dei valori.

```
virtual int  DrawScaleBottom(
  const bool  draw,  // flag
  )
```

Parametri

draw

[in] Flag che indica se la scala deve essere ridisegnata.

Valore di ritorno

Altezza della scala dei valori.

DrawGrid

Metodo virtuale per la ridefinizione della griglia.

```
virtual void  DrawGrid()
```

DrawChart

Metodo virtuale per la ridefinizione del grafico.

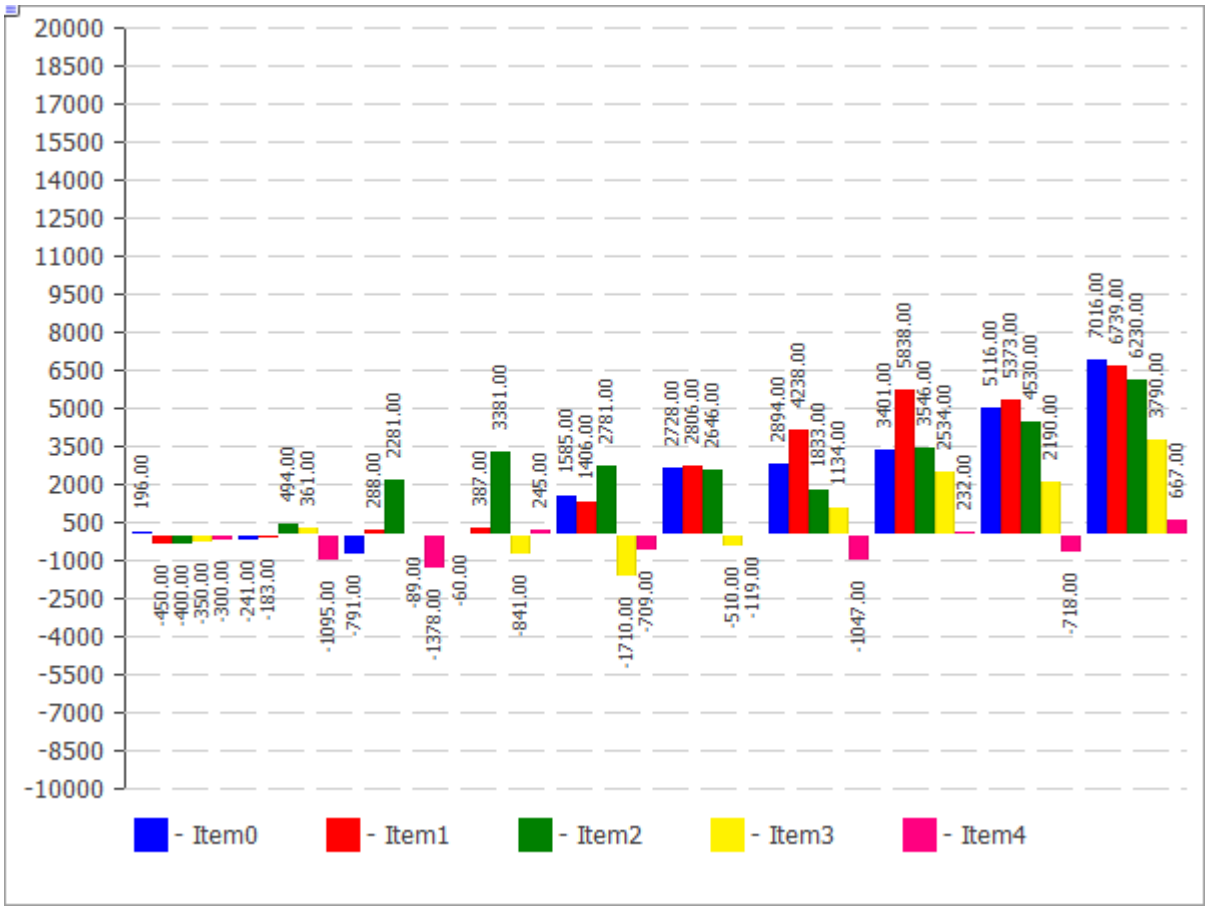
```
virtual void  DrawChart()
```

CHistogramChart

Classe per l'analisi di istogrammi.

Descrizione

Tutti i metodi per lavorare con la compilazione di istogrammi sono implementati in questa classe. Possono essere utilizzati per impostare la larghezza della colonna e per configurare il lavoro con la serie di dati. Sono inclusi i metodi per il lavoro con il riempimento gradiente di colonne di istogramma, che consentono di visualizzare in modo più chiaro i dati.



Il codice della figura precedente è fornito sotto .

Dichiarazione

```
class CHistogramChart : public CChartCanvas
```

Titolo

```
#include <Canvas\Charts\HistogramChart.mqh>
```

Gerarchia ereditaria

CCanvas

CChartCanvas

CHistogramChart

Metodi di classe

Metodo	Azione
Gradient	Imposta la flag che indica se il riempimento gradiente delle colonne dell'istogramma verrà applicato.
BarGap	Imposta il valore dell'offset dell'istogramma dall'origine.
BarMinSize	Imposta la larghezza minima delle colonne dell'istogramma.
BarBorder	Imposta la flag che indica la necessità di disegnare il bordo per ogni colonna.
Create	Metodo virtuale che crea una risorsa grafica.
SeriesAdd	Aggiunge una nuova serie di dati.
SeriesInsert	Inserisce serie di dati nel grafico.
SeriesUpdate	Aggiorna serie di dati sul grafico.
SeriesDelete	Elimina serie di dati dal grafico.
ValueUpdate	Aggiorna il valore dell'elemento nella serie specificata.
DrawData	Metodo virtuale che traccia un istogramma per la serie specificata.
DrawBar	Disegna una colonna dell'istogramma come un rettangolo pieno.
GradientBrush	Crea un pennello per il riempimento gradiente.

Metodi ereditati da CCanvas

Create, CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Metodi ereditati da CChartCanvas

Create, ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Esempio

```
//+------------------------------------------------------------------+
//|                                          HistogramChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Esempio d'uso dell'istogramma"
//---
#include <Canvas\Charts\HistogramChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=true;
//+------------------------------------------------------------------+
//| Funzione start programma script                                  |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- create chart
   CHistogramChart chart;
   if(!chart.CreateBitmapLabel("SampleHistogramChart",10,10,600,450))
     {
      Print("Error creating histogram chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,20);
   chart.ShowValue(true);
   chart.ShowScaleTop(false);
   chart.ShowScaleBottom(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- gioca con i valori
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

Gradient

Imposta la flag che indica se il riempimento gradiente delle colonne dell'istogramma verrà applicato.

```
void  Gradient(
  const bool  flag=true,  // valore flag
  )
```

Parametri

flag=true

Valore flag: true se il riempimento gradiente è abilitato, altrimenti — false.

BarGap

Imposta il valore dell'offset dell'istogramma dall'origine.

```
void  BarGap(
  const uint  value,  // offset
  )
```

Parametri

valore

[in] Valore dell'offset dell'istogramma.

BarMinSize

Imposta la larghezza minima delle colonne dell'istogramma.

```
void  BarMinSize(
  const uint  value,  // spessore minimo
  )
```

Parametri

valore

[in] La larghezza minima.

BarBorder

Imposta la flag che indica la necessità di disegnare il bordo per ogni colonna.

```
void  BarBorder(
  const uint  value,  // flag
  )
```

Parametri

valore

[in] Valore flag:

- true — i bordi verranno disegnati
- false — i bordi non saranno disegnati

Create

Metodo virtuale che crea una risorsa grafica.

```
virtual bool  Create(
  const string        name,    // nome
  const int           width,   // spessore
  const int           height,  // altezza
  ENUM_COLOR_FORMAT   clrfmt,  // formato
  )
```

Parametri

name

[in] Basi per un nome di risorsa grafica. Un nome di risorsa viene generato durante la creazione aggiungendo una stringa pseudocasuale.

width

[in] Larghezza (dimensione lungo l'asse x) in pixel.

height

[in] Altezza (dimensione lungo l'asse Y) in pixel.

clrfmt

[in] Metodo di elaborazione del colore. Vedere la descrizione della funzione ResourceCreate() per ulteriori informazioni sui metodi di elaborazione dei colori.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesAdd

Aggiunge una nuova serie di dati.

```
bool  SeriesAdd(
  const double&  value[],  // valori
  const string   descr,    // etichetta
  const uint     clr,      // colore
  )
```

Parametri

value[]

[In] Serie di Dati.

descr

[in] Etichetta della serie.

clr

[In] Colore di visualizzazione della serie.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesInsert

Inserisce serie di dati nel grafico.

```
bool  SeriesInsert(
  const uint    pos,        // indice
  const double&  value[],   // valori
  const string  descr,      // etichetta
  const uint    clr,        // colore
  )
```

Parametri

pos

[in] Indice per inserimento.

value[]

[In] Serie di Dati.

descr

[in] Etichetta della serie.

clr

[In] Colore di visualizzazione della serie.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesUpdate

Aggiornamenti serie di dati sul grafico.

```
bool  SeriesUpdate(
  const uint    pos,        // indice
  const double  &value[],   // valore
  const string  descr,      // etichetta
  const uint    clr,        // colore
  )
```

Parametri

pos

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

&value[]

[in] Nuovi valori per la serie di dati.

descr

[in] Etichetta della serie.

clr

[In] Colore di visualizzazione della serie.

Valore di ritorno

true se riuscito, altrimenti false.

SeriesDelete

Elimina serie di dati dal grafico.

```
bool  SeriesDelete(
  const uint  pos,  // indice
  )
```

Parametri

pos

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

Valore di ritorno

true se riuscito, altrimenti — false.

ValueUpdate

Aggiorna il valore specificato nella serie specificata.

```
bool  ValueUpdate(
  const uint  series,  // indice delle serie
  const uint  pos,     // indice degli elementi
  double      value,   // valore
  )
```

Parametri

series

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

pos

[in] Indice di elemento della serie.

valore

[in] Nuovo valore.

Valore di ritorno

true se riuscito, altrimenti — false.

DrawData

Metodo virtuale che traccia un istogramma per la serie specificata.

```
virtual void  DrawData(
  const uint  index,  // indice
  )
```

Parametri

index

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

DrawBar

Disegna una colonna dell'istogramma come un rettangolo pieno.

```
void  DrawBar(
  const int   x,    // coordinate X
  const int   y,    // coordinate Y
  const int   w,    // spessore
  const int   h,    // altezza
  const uint  clr,  // colore
  )
```

Parametri

x

[in] coordinate X del punto superiore sinistro del rettangolo . </ T1>

y

[in] coordinate Y del punto in alto a sinistra del rettangolo.

w

[in] La larghezza del rettangolo.

h

[in] L'altezza del rettangolo.

clr

[in] Il colore del rettangolo.

GradientBrush

Crea un pennello per il riempimento gradiente.

```
void  GradientBrush(
  const int   size,      // grandezza
  const uint  fill_clr,  // riempi colore
  )
```

Parametri

size

[in] Spessore del pennello

fill_clr

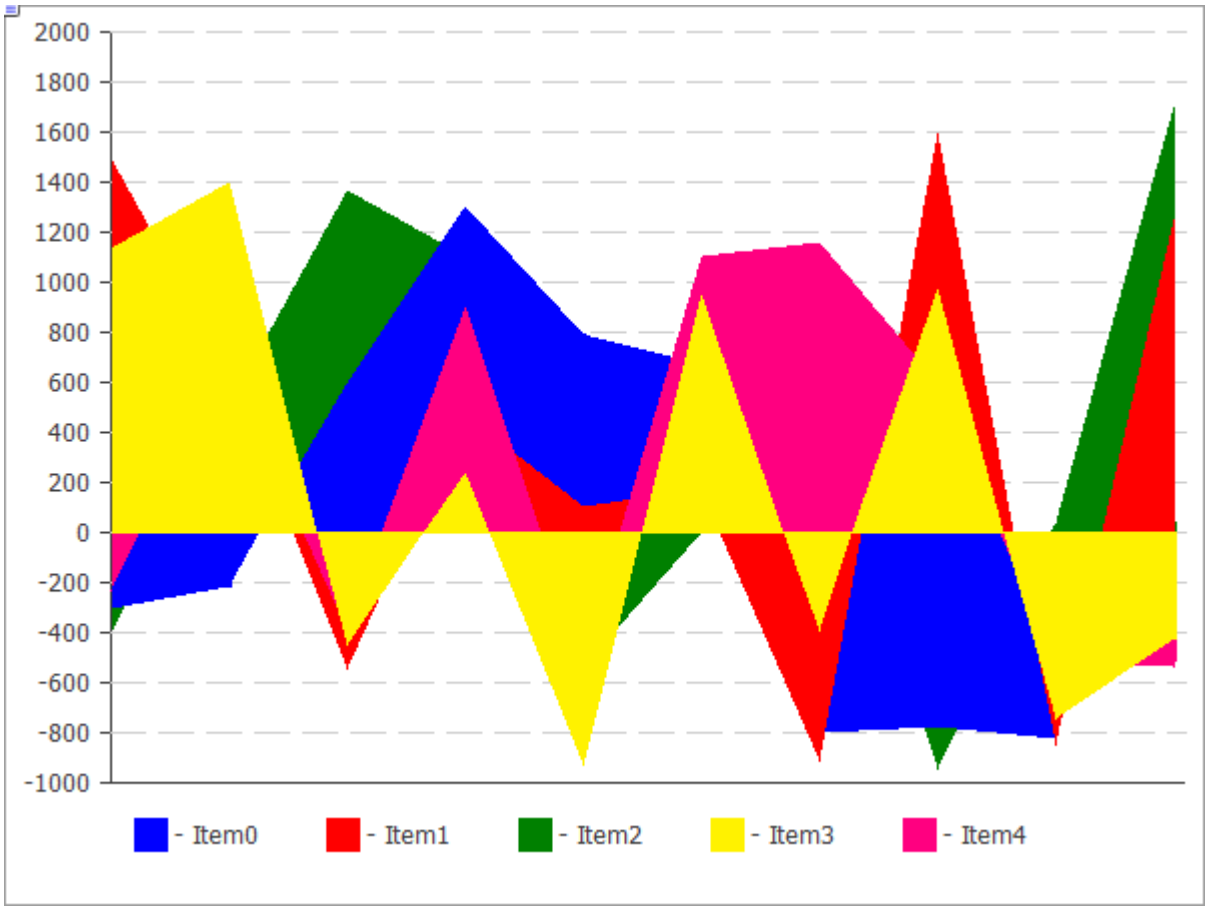
[in] Riempimento colore

CLineChart

Una classe per tracciare curve.

Descrizione

I metodi inclusi in questa classe sono progettati per lavorare con curve sul chart. Ha la capacità di riempire l'area limitata dalla curva di disegno.



Il codice della figura precedente è fornito sotto .

Dichiarazione

```
class CLineChart : public CChartCanvas
```

Titolo

```
#include <Canvas\Charts\LineChart.mqh>
```

Gerarchia ereditaria

CCanvas

CChartCanvas

CLineChart

Metodi di classe

Metodo	Azione
Filled	Imposta la flag per riempire l'area sotto la curva definita dalla serie di dati.
Create	Crea una risorsa grafica.
SeriesAdd	Aggiunge una nuova serie di dati.
SeriesInsert	Inserisce serie di dati nel grafico.
SeriesUpdate	Aggiorna serie di dati sul grafico.
SeriesDelete	Elimina serie di dati dal grafico.
ValueUpdate	Aggiorna il valore specificato nella serie specificata.
DrawChart	Metodo virtuale che disegna una curva e tutti i suoi elementi.
DrawData	Metodo virtuale che disegna una curva per la serie specificata.
CalcArea	Calcola l'area sotto la curva definita dalla serie di dati.

Metodi ereditati da CCanvas

Create, CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Metodi ereditati da CChartCanvas

Create, ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Esempio

```
//+------------------------------------------------------------------+
//|                                                LineChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Esempio di uso della linea chart"
//---
#include <Canvas\Charts\LineChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=false;
//+------------------------------------------------------------------+
//| Funzione start programma script                                  |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- crea chart
   CLineChart chart;
//--- crea chart
   if(!chart.CreateBitmapLabel("SampleHistogrammChart",10,10,600,450))
     {
      Print("Errore creazione linea chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,15);
   chart.ShowScaleTop(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   chart.Filled();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- play with values
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

Filled

Imposta la flag che indica se è necessario riempire l'area sotto la curva definita dalla serie di dati.

```
void  Filled(
  const bool  flag=true,  // flag
  )
```

Parametri

flag=true

[in] Valore flag:

- true — riempire l'area sotto la curva
- false — non riempire l'area sotto la curva

Create

Metodo virtuale che crea una risorsa grafica.

```
virtual bool  Create(
  const string        name,    // nome
  const int           width,   // spessore
  const int           height,  // altezza
  ENUM_COLOR_FORMAT   clrfmt,  // formato
  )
```

Parametri

name

[in] Basi per un nome di risorsa grafica. Un nome di risorsa viene generato durante la creazione aggiungendo una stringa pseudocasuale.

width

[in] Larghezza (dimensione lungo l'asse x) in pixel.

height

[in] Altezza (dimensione lungo l'asse Y) in pixel.

clrfmt

[in] Metodo di elaborazione del colore. Vedere la descrizione della funzione ResourceCreate() per ulteriori informazioni sui metodi di elaborazione dei colori.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesAdd

Aggiunge una nuova serie di dati.

```
bool  SeriesAdd(
  const double&  value[],  // valori
  const string   descr,    // etichetta
  const uint     clr,      // colore
  )
```

Parametri

value[]

[In] Serie di Dati.

descr

[in] Etichetta della serie.

clr

[In] Colore di visualizzazione della serie.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesInsert

Inserisce serie di dati nel grafico.

```
bool  SeriesInsert(
  const uint     pos,        // indice
  const double&  value[],    // valori
  const string   descr,      // etichetta
  const uint     clr,        // colore
  )
```

Parametri

pos

[in] Indice per inserimento.

value[]

[In] Serie di Dati.

descr

[in] Etichetta della serie.

clr

[In] Colore di visualizzazione della serie.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesUpdate

Aggiornamenti serie di dati sul grafico.

```
bool  SeriesUpdate(
  const uint     pos,        // indice
  const double&  value[],    // valori
  const string   descr,      // etichetta
  const uint     clr,        // colore
  )
```

Parametri

pos

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

value[]

[in] Nuovi valori per la serie di dati.

descr

[in] Etichetta della serie.

clr

[In] Colore di visualizzazione della serie.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesDelete

Elimina serie di dati dal grafico.

```
bool  SeriesDelete(
  const uint  pos,  // indice
  )
```

Parametri

pos

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

Valore di ritorno

true se riuscito, altrimenti — false.

ValueUpdate

Aggiorna il valore specificato nella serie specificata.

```
bool  ValueUpdate(
  const uint  series,  // indice delle serie
  const uint  pos,     // indice degli elementi
  double      value,   // valore
  )
```

Parametri

series

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

pos

[in] Indice di elemento della serie.

valore

[in] Nuovo valore.

Valore di ritorno

true se riuscito, altrimenti — false.

DrawChart

Metodo virtuale che disegna una curva e tutti i suoi elementi.

```
virtual void  DrawChart()
```

DrawData

Metodo virtuale che disegna una curva per la serie specificata.

```
virtual void  DrawData(
  const uint  index,  // indice
  )
```

Parametri

index

[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

CalcArea

Calcola l'area sotto la curva definita dalla serie di dati.

```
double  CalcArea(
  const uint  index,  // indice
  )
```

Parametri

index

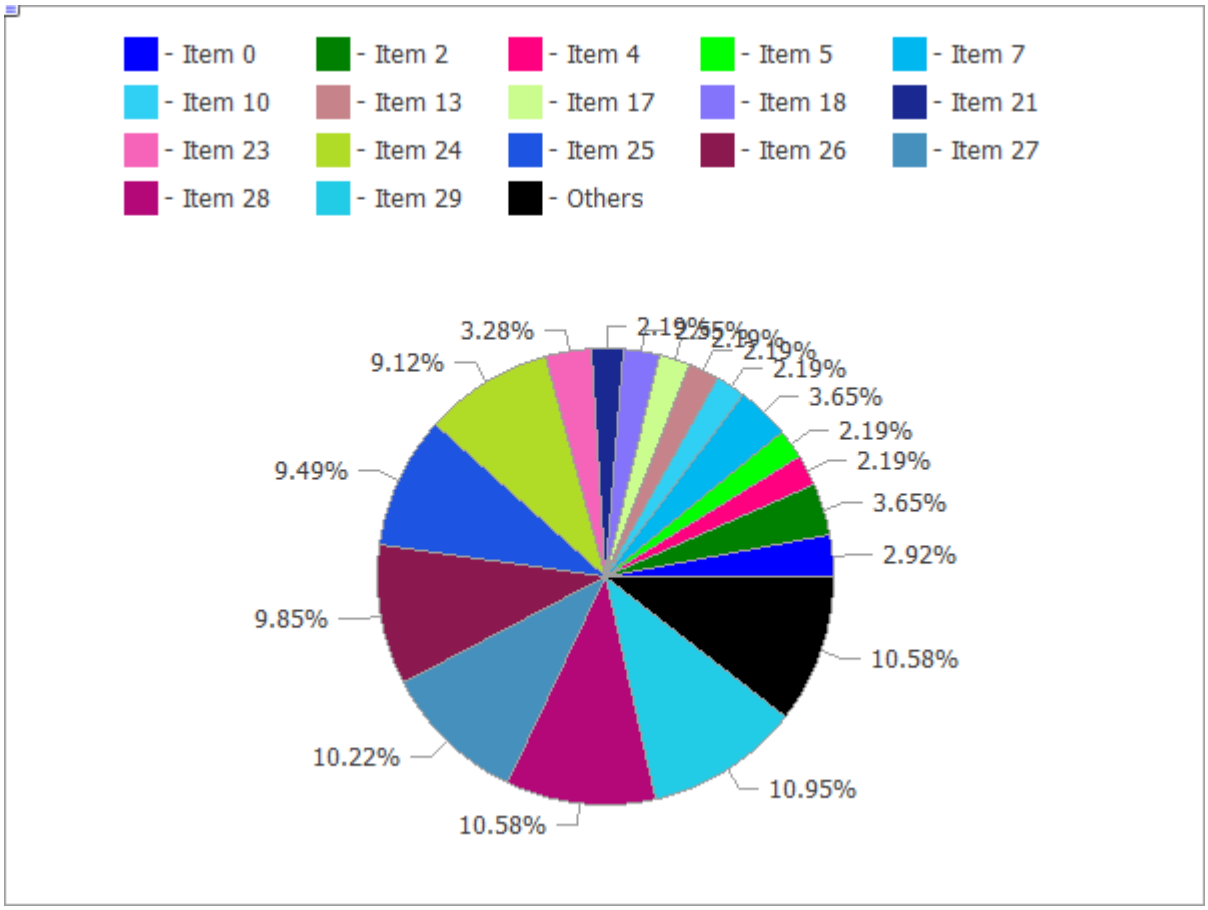
[in] Indice della serie — il numero di serie della sua aggiunta, a partire da 0.

Valore di ritorno

Area della figura limitata dalla curva definita dalla serie di dati.

CPieChart

Classe per la compilazione di grafici a torta.



Il codice della figura precedente è fornito sotto .

Descrizione

I metodi inclusi in questa classe sono progettati per operare su scala completa con grafici a torta, dalla creazione di una risorsa grafica alla progettazione di etichette, a segmenti.

Dichiarazione

```
class CPieChart : public CChartCanvas
```

Titolo

```
#include <Canvas\Charts\PieChart.mqh>
```

Gerarchia ereditaria

CCanvas

CChartCanvas

CPieChart

Metodi di classe

Metodo	Azione
Create	Metodo virtuale che crea una risorsa grafica.
SeriesSet	Imposta una serie di valori che saranno mostrati sul grafico a torta.
ValueAdd	Aggiunge un nuovo valore al grafico a torta (alla fine).
ValueInsert	Inserisce un nuovo valore nel grafico a torta (alla posizione specificata).
ValueUpdate	Aggiorna il valore sul grafico a torta (nella posizione specificata).
ValueDelete	Rimuove un valore dal grafico a torta (nella posizione specificata).
DrawChart	Metodo virtuale che disegna un grafico a torta e tutti i suoi elementi.
DrawPie	Disegna un segmento del grafico a torta, che corrisponde a un valore specificato.
LabelMake	Genera un'etichetta di segmento in base al suo valore e all'etichetta originale.

Metodi ereditati da CCanvas

Create, CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Metodi ereditati da CChartCanvas

Create, ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Esempio

```
//+------------------------------------------------------------------+
//|                                                PieChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Esempio di utilizzo del grafico a torta"
//---
#include <Canvas\Charts\PieChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input int      Width=600;
input int      Height=450;
//+------------------------------------------------------------------+
//| Funzione di avvio del programma script.                          |
//+------------------------------------------------------------------+
int OnStart(void)
  {
//--- check
   if(Width<=0 || Height<=0)
     {
     Print("Troppo semplice.");
      return(-1);
     }
//--- crea chart
   CPieChart pie_chart;
   if(!pie_chart.CreateBitmapLabel("PieChart",10,10,Width,Height))
     {
     Print("Errore nella creazione del grafico a torta: ",GetLastError());
      return(-1);
     }
   pie_chart.ShowPercent();
//--- disegna
   for(uint i=0;i<30;i++)
     {
      pie_chart.ValueAdd(100*(i+1),"Item "+IntegerToString(i));
      Sleep(10);
     }
   Sleep(2000);
//--- disabilita legenda
   pie_chart.LegendAlignment(ALIGNMENT_LEFT);
   Sleep(2000);
//--- disabilita legenda
   pie_chart.LegendAlignment(ALIGNMENT_RIGHT);
   Sleep(2000);
```

```
//--- disabilita legenda
   pie_chart.LegendAlignment(ALIGNMENT_TOP);
   Sleep(2000);
//--- disabilita legenda
   pie_chart.ShowLegend(false);
   Sleep(2000);
//--- disabilita percentuale
   pie_chart.ShowPercent(false);
   Sleep(2000);
//--- disabilita descrittori
   pie_chart.ShowDescriptors(false);
   Sleep(2000);
//--- abilita tutto
   pie_chart.ShowLegend();
   pie_chart.ShowValue();
   pie_chart.ShowDescriptors();
   Sleep(2000);
//--- o come questo
   pie_chart.ShowFlags(FLAG_SHOW_LEGEND|FLAG_SHOW_DESCRIPTORS|FLAG_SHOW_PERCENT);
   uint total=pie_chart.DataTotal();
//--- gioca con i valori
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ValueUpdate(i,100*(rand()%10+1));
      Sleep(1000);
     }
//--- gioca con i colori
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ColorUpdate(i%total,RandomRGB());
      Sleep(1000);
     }
//--- ruota
   while(!IsStopped())
     {
      pie_chart.DataOffset(pie_chart.DataOffset()+1);
      Sleep(200);
     }
//--- fine
   pie_chart.Destroy();
   return(0);
  }
//+------------------------------------------------------------------+
//| Colori RGB casuali                                               |
//+------------------------------------------------------------------+
uint RandomRGB(void)
  {
   return(XRGB(rand()%255,rand()%255,rand()%255));
  }
```

Create

Metodo virtuale che crea una risorsa grafica.

```
virtual bool  Create(
  const string        name,    // nome
  const int           width,   // spessore
  const int           height,  // altezza
  ENUM_COLOR_FORMAT   clrfmt,  // formato
  )
```

Parametri

name

[in] Basi per un nome di risorsa grafica. Un nome di risorsa viene generato durante la creazione aggiungendo una stringa pseudocasuale.

width

[in] Larghezza (dimensione lungo l'asse x) in pixel.

height

[in] Altezza (dimensione lungo l'asse Y) in pixel.

clrfmt

[in] Metodo di elaborazione del colore. Vedere la descrizione della funzione ResourceCreate() per ulteriori informazioni sui metodi di elaborazione dei colori.

Valore di ritorno

true se riuscito, altrimenti — false.

SeriesSet

Imposta una serie di valori che saranno mostrati sul grafico a torta.

```
bool  SeriesSet(
  const double&  value[],  // valori
  const string&  text[],   // etichetta
  const uint&    clr[],    // colore
  )
```

Parametri

value[]

[in] Array dei valori.

text[]

[in] Array dei valori dell'etichetta.

clr[]

[in] Array dei valori dei colori.

Valore di ritorno

true se riuscito, altrimenti — false.

ValueAdd

Aggiunge un nuovo valore al grafico a torta (alla fine).

```
bool  ValueAdd(
  const double  value,  // valore
  const string  descr,  // etichetta
  const uint    clr,    // colore
  )
```

Parametri

valore

[in] Valore.

descr

[in] Valore etichetta.

clr

[in] Value del colore.

Valore di ritorno

true se riuscito, altrimenti — false.

ValueInsert

Inserisce un nuovo valore nel grafico a torta (alla posizione specificata).

```
bool  ValueInsert(
  const uint    pos,    // indice
  const double  value,  // valore
  const string  descr,  // etichetta
  const uint    clr,    // colore
  )
```

Parametri

pos

[in] Indice per inserimento.

valore

[in] Valore.

descr

[in] Valore etichetta.

clr

[in] Value del colore.

Valore di ritorno

true se riuscito, altrimenti — false.

ValueUpdate

Aggiorna il valore sul grafico a torta (nella posizione specificata).

```
bool  ValueUpdate(
  const uint    pos,    // indice
  const double  value,  // valore
  const string  descr,  // etichetta
  const uint    clr,    // colore
  )
```

Parametri

pos

[in] Indice del valore — il numero di serie della sua aggiunta, a partire da 0.

valore

[in] Valore.

descr

[in] Valore etichetta.

clr

[in] Value del colore.

Valore di ritorno

true se riuscito, altrimenti — false.

ValueDelete

Rimuove un valore dal grafico a torta (nella posizione specificata).

```
bool  ValueDelete(
  const uint  pos,  // indice
  )
```

Parametri

pos

[in] Indice del valore — il numero di serie della sua aggiunta, a partire da 0.

Valore di ritorno

true se riuscito, altrimenti — false.

DrawChart

Metodo virtuale che disegna un grafico a torta e tutti i suoi elementi.

```
virtual void  DrawChart()
```

DrawPie

Disegna un segmento del grafico a torta, che corrisponde a un valore specificato.

```
void  DrawPie(
  double      fi3,   // angolo del raggio dal centro della torta, che definisce il pr
  double      fi4,   // angolo del raggio dal centro della torta, che definisce il se
  int         idx,   // indice
  CPoint&     p[],  //
  const uint  clr,   //
  )
```

Parametri

fi3

[in] Angolo in radianti, che definisce il primo limite dell'arco.

fi4

[in] Angolo in radianti, che definisce il secondo limite dell'arco.

idx

[in] Indice del valore corrispondente al segmento.

p[]

[in] Array di punti di riferimento (x, y) per la compilazione dei segmenti.

clr

[in] Colore del segmento.

LabelMake

Genera un'etichetta di segmento in base al suo valore e all'etichetta originale.

```
string  LabelMake(
  const string  text,     // etichetta
  const double  value,    // valore
  const bool    to_left,  // flag
  )
```

Parametri

text

[in] Etichetta.

valore

[in] Valore.

to_left

[in] Definisce l'ordine del layout dell'etichetta:

- true — etichetta, quindi valore.
- false — valore, quindi etichetta.

Valore di ritorno

Etichetta del segmento.

cGrafica 3D

La sezione presenta le classi per lo sviluppo della grafica tridimensionale. Le classi si basano sulle funzioni per lavorare con le DirectX. CCnavas3D è una classe di base contenente i metodi per la gestione della videocamera e dell'illuminazione, oltre a presentare il gestore delle risorse grafiche - textures, shaders, vertex buffers, indici e parametri dello shader.

Inoltre, ci sono le classi degli oggetti base della scena, come un box, una superficie tridimensionale sui dati dell'utente, ed una griglia arbitraria.

CCanvas3D

CCanvas3D è una classe per la creazione e la visualizzazione semplificate di oggetti 3D su un chart.

Descrizione

CCanvas3D semplifica notevolmente la creazione e la visualizzazione di grandi quantità di dati sotto forma di grafica 3D animata. La classe contiene i metodi per la gestione della telecamera e dell'illuminazione, oltre a funzioni di gestione delle risorse per la creazione di risorse grafiche: textures, shaders, vertex buffers, indici e parametri shaders.

Inoltre, la libreria contiene le classi degli oggetti base della scena, come un box, una superficie tridimensionale sui dati dell'utente, ed una griglia arbitraria.

La scheda video dovrebbe supportare DX 11 e Shader Model 5.0 affinché le funzioni girino.

Dichiarazione

```
class CCanvas
```

Titolo

```
#include <Canvas\Canvas.mqh>
```

Gerarchia dell'ereditarietà

CCanvas

CCanvas3D

Metodi della classe per gruppi

Creazione/eliminazione	Descrizione
Create	Crea una risorsa grafica per il rendering di una scena 3D senza associazione ad un oggetto chart.
Attach	Ottiene una risorsa grafica dall'oggetto OBJ_BITMAP_LABEL e la collega ad un'istanza della classe CCanvas.
ObjectAdd	Aggiunge un oggetto ad una scena 3D per il rendering successivo.
Destroy	Distrugge una risorsa grafica e rilascia un contesto 3D grafico.
Light	
AmbientColorSet	Imposta il colore e l'intensità dell'illuminazione generale a tutto tondo.
AmbientColorGet	Ottiene il colore e l'intensità dell'illuminazione a tutto tondo ambientale.
LightDirectionSet	Imposta la direzione di una sorgente luminosa diretta.
LightDirectionGet	Ottiene la direzione di una sorgente luminosa diretta.

Creazione/eliminazione	Descrizione
LightColorSet	Imposta il colore e l'intensità di una fonte di luce diretta.
LightColorGet	Ottiene il colore e l'intensità di una sorgente luminosa diretta.
Camera	
ProjectionMatrixSet	Calcola ed imposta una matrice di proiezione delle coordinate 3D su un frame 2D.
ProjectionMatrixGet	Ottiene una matrice di proiezione di scene 3D su un frame 2D.
ViewMatrixSet	Imposta una matrice di visualizzazione scena 3D.
ViewMatrixGet	Restituisce una matrice di visualizzazione scena 3D.
ViewPositionSet	Imposta il punto di vista sulla scena 3D.
ViewRotationSet	Imposta la direzione dello sguardo sulla scena 3D.
ViewTargetSet	Imposta le coordinate del punto a cui è diretto lo sguardo.
ViewUpDirectionSet	Imposta la direzione del bordo superiore del riquadro nello spazio 3D.
Rendering	
Render	Esegue il rendering di tutti gli oggetti della scena nel buffer interno del frame per la visualizzazione successiva.
RenderBegin	Prepara un contesto grafico per il rendering di un nuovo frame.
RenderEnd	Copia un frame renderizzato nel buffer interno ed aggiorna l'immagine del chart, se necessario.
Ottenere risorse	
DXContext	Ottiene l'handle del contesto grafico.
DXDispatcher	Ottiene l'handle del dispatcher di risorse.
InputScene	Ottiene il puntatore al buffer dei parametri della scena.

AmbientColorGet

Ottiene il colore e l'intensità dell'illuminazione a tutto tondo ambientale.

```
void  AmbientColorGet(
   DXColor  &ambient_color      // colore e intensità della luce a 360 gradi
   );
```

Parametri

&ambient_color

[out] Colore di illuminazione a tutto tondo.

Valore di Ritorno

No

Nota

L'intensità è memorizzata nel canale alfa della struttura DXColor.

AmbientColorSet

Imposta il colore e l'intensità dell'illuminazione generale a tutto tondo.

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // colore ed intensità della luce a 360 gradi
   );
```

Parametri

&ambient_color

[in] Colore di illuminazione a tutto tondo.

Valore di Ritorno

No

Nota

L'intensità è impostata nel canale alfa della struttura DXColor.

Attach

Ottiene la risorsa grafica da un oggetto OBJ_BITMAP_LABEL e lo collega ad un'istanza della classe CCanvas.

```
bool  Attach(
   const long          chart_id,                              // ID chart
   const string        objname,                               // nome oggetto
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // metodo di gestione del
   )
```

Crea una risorsa grafica per un oggetto OBJ_BITMAP_LABEL e lo collega ad un'istanza della classe CCanvas.

```
bool  Attach(
   const long          chart_id,                              // ID chart
   const string        objname,                               // nome oggetto
   const int           width,                                 // larghezza dell'immagi
   const int           height,                                // altezza dell'immagine
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // metodo di gestione del
   )
```

Parametri

chart_id

[in] ID Chart.

objname

[in] Nome dell'oggetto grafico.

width

[in] Larghezza del frame in una risorsa.

height

[in] Altezza del frame.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Metodo di gestione del colore. Vedere la descrizione della funzione ResourceCreate() per saperne di più sui metodi di gestione del colore.

Nota

true - in caso di successo, false - se non è stato possibile aggiungere un oggetto grafico.

Create

Crea una risorsa grafica per il rendering di una scena 3D senza associazione ad un oggetto chart.

```
virtual bool  Create(
   const string       name,                              // nome dell'oggetto grafi
   const int          width,                             // spessore
   const int          height,                            // altezza
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // formato colore
   );
```

Parametri

name

[in] Nome dell'oggetto grafico.

width

[in] Larghezza del frame.

height

[in] Altezza del frame.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Metodo di gestione del colore. Vedere la descrizione della funzione ResourceCreate() per saperne di più sui metodi di gestione del colore.

Nota

true - se viene creata una risorsa, altrimenti - false.

Destroy

Distrugge una risorsa grafica e rilascia un contesto 3D grafico.

```
virtual void  Destroy()
```

Valore di Ritorno

No

Nota

Se una risorsa grafica è stata associata ad un oggetto grafico, quest'ultimo viene eliminato.

DXContext

Ottiene l'handle del contesto grafico.

```
int  DXContext()
```

Valore di Ritorno

Handle del contesto grafico.

DXDispatcher

Ottiene l'handle del dispatcher di risorse.

```
CDXDispatcher*  DXDispatcher()
```

Valore di Ritorno

Handle del dispatcher di risorse.

InputScene

Ottiene il puntatore al buffer dei parametri della scena.

```
CDXInput*  InputScene()
```

Valore di Ritorno

Puntatore al buffer dei parametri della scena.

LightColorGet

Ottiene il colore e l'intensità di una sorgente luminosa diretta.

```
void  LightColorGet(
   DXColor  &light_color      // colore ed intensità dell'illuminazione direzionale
   );
```

Parametri

&light_color

[out] Colore e intensità dell'illuminazione direzionale.

Valore di Ritorno

Nessuno.

Nota

L'intensità è memorizzata nel canale alfa della struttura DXColor.

LightColorSet

Imposta il colore e l'intensità di una fonte di luce diretta.

```
void  LightColorSet(
   const DXColor  &light_color      // colore ed intensità dell'illuminazione direzio
   );
```

Parametri

&light_color

[in] Colore ed intensità dell'illuminazione direzionale.

Valore di Ritorno

Nessuno.

Nota

L'intensità è impostata nel canale alfa della struttura DXColor.

LightDirectionGet

Ottiene la direzione di una sorgente luminosa diretta.

```
void  LightDirectionGet(
   DXVector3  &light_direction      // vettore di direzione
   );
```

Parametri

&light_direction

[out] Vettore di direzione.

Valore di Ritorno

Nessuno.

LightDirectionSet

Imposta la direzione di una sorgente luminosa diretta.

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // vettore di direzione
   );
```

Parametri

&light_direction

[in] Vettore di direzione.

Valore di Ritorno

Nessuno.

ObjectAdd

Aggiunge un oggetto ad una scena 3D per il rendering successivo.

```
bool  ObjectAdd(
   CDXObject  *object      // puntatore all'oggetto
   );
```

Parametri

object

[in] Puntatore ad un'istanza della classe derivata dalla classe astratta CDXObject.

Valore di Ritorno

true - in caso di successo, false - se non è stato possibile aggiungere un oggetto grafico 3D.

ProjectionMatrixGet

Ottiene una matrice di proiezione di scene 3D su un frame 2D.

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // matrice di proiezione
   );
```

Parametri

&projection_matrix

[out] Matrice di proiezione.

Valore di Ritorno

Nessuno.

ProjectionMatrixSet

Calcola ed imposta una matrice di proiezione delle coordinate 3D su un frame 2D.

```
void  ProjectionMatrixSet(
   float  fov,              // campo visivo
   float  aspect_ratio,     // rapporto aspetto del frame
   float  z_near,           //
   float  z_far             //
   );
```

Parametri

fov

[in] Larghezza del campo visivo in radianti per creare una proiezione della scena.

aspect_ratio

[in] Rapporto dell'aspetto del frame 2D.

z_near

[in] Distanza dal piano di ritaglio vicino.

z_far

[in] Distanza dal piano di ritaglio lontano.

Valore di Ritorno

Nessuno.

Nota

Il frame 2D mostra solo le proiezioni di oggetti 3D che cadono nel campo visivo specificato e si trovano tra i piani di ritaglio vicino e lontano.

Render

Esegue il rendering di tutti gli oggetti della scena nel buffer interno del frame per la visualizzazione successiva.

```
bool  Render(
   uint  flags,                  // combinazione di flags
   uint  background_color=0      // colore di sfondo
   );
```

Parametri

flags

[in] Combinazione di flag che imposta la modalità di rendering. Valori possibili:
DX_CLEAR_COLOR - cancella il buffer dell'immagine usando*colore di sfondo*.
DX_CLEAR_DEPTH - cancella il buffer di profondità.

background_color=0

[in] colore di sfondo della scena 3D.

Valore di Ritorno

true - in caso di successo, false - se non è stato possibile eseguire il rendering.

Nota

La chiamata di Render() non aggiorna la scena sul chart. Invece, aggiorna solo il buffer interno dell'immagine. Il metodo Update() deve essere chiamato esplicitamente per eseguire il rendering del frame aggiornato.

Render() include le chiamate RenderBegin() e Renderend().

RenderBegin

Prepara un contesto grafico per il rendering di un nuovo frame.

```
virtual bool  RenderBegin(
   uint  flags,                  // combinazione di flags
   uint  background_color=0      // colore di sfondo
   );
```

Parametri

flags

[in] Combinazione di flag che imposta la modalità di rendering. Valori possibili:
DX_CLEAR_COLOR - cancella il buffer dell'immagine usando*colore di sfondo*.
DX_CLEAR_DEPTH - cancella il buffer di profondità.

background_color=0

[in] colore di sfondo della scena 3D.

Valore di Ritorno

true - in caso di successo, false - se non è stato possibile aggiornare gli input dello shader.

RenderEnd

Copia un frame renderizzato nel buffer interno ed aggiorna l'immagine del chart, se necessario.

```
virtual bool  RenderEnd(
   bool  redraw=false      // aggiorna flag
   );
```

Parametri

redraw=false

[in] Flag di necessità di ridisegno del chart.

Valore di Ritorno

true - successo, altrimenti - false.

ViewMatrixGet

Restituisce una matrice di visualizzazione scena 3D.

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // visualizza matrice
   );
```

Parametri

&view_matrix

[out] Visualizza la matrice che imposta la posizione e la direzione della telecamera nello spazio 3D.

Valore di Ritorno

Nessuno.

ViewMatrixSet

Imposta una matrice di visualizzazione scena 3D.

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // visualizza matrice
   );
```

Parametri

&view_matrix

[in] Visualizza la matrice che imposta la posizione e la direzione della telecamera nello spazio 3D.

Valore di Ritorno

Nessuno.

ViewPositionSet

Imposta il punto di vista sulla scena 3D.

```
void  ViewPositionSet(
   const DXVector3  &position      // posizione del punto di vista
   );
```

Parametri

&position

[in] Impostazione della posizione di un punto di vista sulla scena 3D.

Valore di Ritorno

Nessuno.

Nota

L'impostazione della posizione del punto di vista utilizzando ViewPositionSet() modifica la matrice della vista ottenuta in ViewMatrixGet().

ViewRotationSet

Imposta la direzione dello sguardo sulla scena 3D.

```
void  ViewRotationSet(
   const DXVector3  &rotation      // vettore di rotazione angoli
   );
```

Parametri

&rotation

[in] Impostazione vettoriale Angoli di Eulero per calcolare la direzione di uno sguardo su una scena 3D.

Valore di Ritorno

Nessuno.

Nota

L'impostazione della direzione dello sguardo tramite ViewRotationSet() modifica la matrice della vista ottenuta in ViewMatrixGet().

ViewTargetSet

Imposta le coordinate del punto a cui è diretto lo sguardo.

```
void  ViewTargetSet(
   const DXVector3  &target      // coordinate target
   );
```

Parametri

&target

[in] Coordinate del punto a cui è diretto uno sguardo.

Valore di Ritorno

Nessuno.

Nota

Utilizzato per fissare lo sguardo in un punto della scena quando si sposta il punto di vista.

L'impostazione di una nuova coordinata di destinazione mediante ViewRotationSet() modifica la matrice della vista ottenuta in ViewMatrixGet().

ViewTargetSet() viene utilizzato insieme a ViewUpDirectionSet() per definire la direzione dello sguardo.

ViewUpDirectionSet

Imposta la direzione del bordo superiore del riquadro nello spazio 3D.

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // direzione superiore
   );
```

Parametri

&up_direction

[in] Direzione della parte superiore del frame nello spazio 3D.

Valore di Ritorno

Nessuno.

Nota

L'impostazione di una nuova direzione mediante ViewUpDirectionSet() modifica la matrice della vista ottenuta in ViewMatrixGet().

ViewUpDirectionSet() viene utilizzato insieme a ViewTargetSet() per definire la direzione dello sguardo.

CChart

CChart è una classe per l'accesso semplificato alle proprietà dell' oggetto grafico "Chart".

Descrizione

La classe cChart fornisce l'accesso alle proprietà dell'oggetto "Chart".

Dichiarazione

```
class CChart : public CObject
```

Titolo

```
#include <Charts\Chart.mqh>
```

Gerarchia di ereditarietà

CObject
  CChart

I Metodi della Classe per Gruppi

L'accesso ai dati protetti	
ChartID	Ottiene identificativo del chart
Proprietà generali	
Mode	Ottiene/Imposta il valore della proprietà "Mode" (barre, candele, o linea)
Foreground	Ottiene/Imposta il valore della proprietà "Foregroud"
Shift	Ottiene/Imposta il valore della proprietà "Shift"
ShiftSize	Ottiene/Imposta il valore della proprietà "ShiftSize" (in percentuale)
AutoScroll	Ottiene/Imposta il valore della proprietà "AutoScroll"
Scale	Ottiene/Imposta il valore della proprietà "Scale"
ScaleFix	Ottiene/Imposta il valore della proprietà "ScaleFix" (scala chart fissa o no)
ScaleFix_11	Ottiene/Imposta il valore della proprietà "ScaleFix_11" (scala chart 1:1, o no)
FixedMax	Ottiene/Imposta il valore della proprietà "FixedMax" (prezzo massimale fissato)
FixedMin	Ottiene/Imposta il valore della proprietà "FixedMin" (prezzo minimale fissato)

L'accesso ai dati protetti	
ScalePPB	Ottiene/Imposta il valore della proprietà "ScalePPB" (la scala è "punto per barra" o no)
PointsPerBar	Ottiene/Imposta il valore della proprietà "PointsPerBar" (in punti per barra)
Mostra proprietà	
ShowOHLC	Ottiene/Imposta il valore della proprietà "ShowOHLC"
ShowLineBid	Ottiene/Imposta il valore della proprietà "ShowLineBid"
ShowLineAsk	Ottiene/Imposta il valore della proprietà "ShowLineAsk"
ShowLastLine	Ottiene/Imposta il valore della proprietà "ShowLastLine"
ShowPeriodSep	Ottiene/Imposta il valore della proprietà "ShowPeriodSep" (mostra separatori periodo)
ShowGrid	Ottiene/Imposta il valore della proprietà "ShowGrid"
ShowVolumes	Ottiene/Imposta il valore della proprietà "ShowVolumes" (colore per i volumi e livelli di posizioni aperte)
ShowObjectDescr	Ottiene/Imposta il valore della proprietà "ShowObjectDescr" (mostra descrizione per oggetti grafici)
ShowDateScale	Ottiene/Imposta il valore della proprietà "ShowDateScale" (scala data del chart)
ShowPriceScale	Ottiene/Imposta il valore della proprietà "ShowPriceScale" (scala prezzo del chart)
Proprietà Colore	
ColorBackground	Ottiene/Imposta il valore della proprietà "ColorBackground" (colore background del chart)
ColorForeground	Ottiene/Imposta il valore della proprietà "ColorForeground" (colore degli assi, scala e stringhe OHLC del chart)
ColorGrid	Ottiene/Imposta il valore della proprietà "ColorGrid" (colore della griglia)
ColorBarUp	Ottiene/Imposta il valore della proprietà "ColorBarUp" (colore per bull bar, la sua ombra e i contorni del corpo della candela)
ColorBarDown	Ottiene/Imposta il valore della proprietà "ColorBarDown" (colore per bear bars, la sua ombra e i contorni del corpo della candela)
ColorCandleBull	Ottiene/Imposta il valore della proprietà "ColorCandleBull" (colore del corpo della candela bull)

L'accesso ai dati protetti	
ColorCandleBear	Ottiene/Imposta il valore della proprietà "ColorCandleBear" (colore del corpo della candela bear)
ColorChartLine	Ottiene/Imposta il valore della proprietà "ColorChartLine" (colore per la linea chart e candele Doji)
ColorVolumes	Ottiene/Imposta il valore della proprietà "ColorVolumes" (colore per volumi e livelli di posizioni aperte)
ColorLineBid	Ottiene/Imposta il valore della proprietà "ColorLineBid" (colore della linea Bid)
ColorLineAsk	Ottiene/Imposta il valore della proprietà "ColorLineAsk" (colore della linea Ask)
ColorLineLast	Ottiene/Imposta il valore della proprietà "ColorLineLast" (colore della linea di prezzo ultimo affare)
ColorStopLevels	Ottiene/Imposta il valore della proprietà "ColorStopLevels" (colore dei livelli di SL e TP)
Proprietà solo lettura	
VisibleBars	Ottiene il numero totale di barre del grafico visibili
WindowsTotal	Ottiene il numero totale di finestre del chart, comprese le sottofinestre dell'indicatore chart
WindowIsVisible	Ottiene la flag di visibilità della sottofinestra chart specificata
WindowHandle	Ottiene l'handle finestra del chart (HWND)
FirstVisibleBar	Ottiene il numero della prima barra visibile del grafico
WidthInBars	Ottiene larghezza della finestra in barre.
WidthInPixels	Ottiene larghezza sottofinestra in pixel.
HeightInPixels	Ottiene l'altezza sottofinestra in pixel.
PriceMin	Ottiene prezzo minimo della sottofinestra specificata
PriceMax	Ottiene prezzo massimo della sottofinestra specificata
Proprietà	
Attach	Assegna il chart corrente all'istanza della classe
FirstChart	Assegna il primo chart del terminale client all'istanza della classe
NextChart	Assegna il prossimo grafico del terminale del client per l'istanza della classe
Open	Apre chart con i parametri specificati e lo assegna all' istanza della classe

L'accesso ai dati protetti	
Detach	Stacca il chart dall' istanza della classe
Close	Chiude il grafico assegnato alla istanza della classe.
BringToTop	Mostra il grafico sopra ad altri chart
EventObjectCreate	Imposta un flag per inviare notifiche di un evento di nuova creazione oggetto su un chart
EventObjectDelete	Imposta un flag per inviare notifiche di un evento di eliminazione oggetto su un grafico
Indicators	
IndicatorAdd	Aggiunge un indicatore con l' handle specificato in una sottofinestra chart specificata
IndicatorDelete	Rimuove un indicatore con un nome specificato dalla sottofinestra chart specificata
IndicatorsTotal	Restituisce il numero di tutti gli indicatori applicati alla sottofinestra chart specificata
IndicatorName	Restituisce il nome breve dell'indicatore sulla sottofinestra chart specificata
Navigazione	
Navigate	Ci si sposta nel chart
L'accesso alle API MQL5	
Symbol	Ottiene simbolo del grafico
Period	Ottiene periodo del grafico
Redraw	Ridisegna il chart, assegnato all'istanza della classe
GetInteger	La funzione restituisce il valore della proprietà dell'oggetto corrispondente
SetInteger	Imposta nuovo valore per la proprietà del tipo integer
GetDouble	La funzione restituisce il valore della proprietà dell'oggetto corrispondente
SetDouble	Imposta nuovo valore per la proprietà di tipo double
GetString	La funzione restituisce il valore della proprietà dell'oggetto corrispondente
SetString	Imposta nuovo valore per la proprietà del tipo string
SetSymbolPeriod	Cambia simbolo e il periodo del chart assegnato alla istanza della classe
ApplyTemplate	Applica il modello specificato al chart

L'accesso ai dati protetti	
ScreenShot	Crea screenshot del grafico specificato e salva in file .gif
WindowOnDropped	Ottiene il numero della sottofinestra chart corrispondente al punto di rilascio dell'oggetto (expert o script)
PriceOnDropped	Ottiene le coordinate prezzo corrispondenti al punto di rilascio dell'oggetto (expert o script)
TimeOnDropped	Ottiene le coordinate tempo corrispondenti al punto di rilascio dell'oggetto (expert o script)
XOnDropped	Ottiene le coordinate X corrispondenti al punto di rilascio dell'oggetto (expert o script)
YOnDropped	Ottiene le coordinate Y corrispondenti al punto di rilascio dell'oggetto (expert o script)
Input/Output	
virtual Save	Salva i parametri degli oggetti su file
virtual Load	Carichi i parametri oggetto da file
virtual Type	Ottiene l'identificatore del tipo di oggetto grafico

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

ChartID

Restituisce l'identificatore del chart.

```
long  ChartID() const
```

Valore di ritorno

Identificatore del Chart assegnato alla istanza della classe del chart. Se non ci sono chart assegnati, restituisce -1.

Mode (Metodo Get)

Ottiene/Imposta il valore della proprietà "Mode" (barre, candele, o linea)

```
ENUM_CHART_MODE  Mode() const
```

Valore di ritorno

Valore della proprietà "Mode" dell'oggetto assegnato all'istanza della classe. Se non ci sono chart assegnati, restituisce WRONG_VALUE.

Mode (Metodo Set)

Imposta il nuovo valore per la proprietà "Mode" (barre, candele o linea).

```
bool  Mode(
   ENUM_CHART_MODE  mode       // modalità chart
   )
```

Parametri

mode

[in] Modalità chart (candele, barre o linea) dell'enumerazione ENUM_CHART_MODE.

Valore di ritorno

true - successo, false - non può cambiare la modalità.

Foreground (Metodo Get)

Ottiene il valore della proprietà "Foreground"(primo piano).

```
bool  Foreground() const
```

Valore di ritorno

Valore della proprietà "Foreground" del chart assegnato alla istanza della classe. Se non c'è chart assegnato, restituisce false.

Foreground (Metodo Set)

Imposta nuovo valore per la proprietà "Foreground".

```
bool  Foreground(
   bool  foreground      // valore flag
   )
```

Parametri

foreground

[in] Nuovo valore per la proprietà "Foreground".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Shift (Metodo Get)

Ottiene il valore della proprietà "Shift".

```
bool  Shift() const
```

Valore di ritorno

Valore della proprietà "Shift" del chart assegnato all'istanza della classe. Se non c'è chart assegnato, restituisce false.

Shift (Metodo Set)

Imposta nuovo valore per la proprietà "Shift".

```
bool  Shift(
   bool  shift      // valore flag
   )
```

Parametri

shift

[in] Nuovo valore per la proprietà "Shift" .

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShiftSize (Metodo Get)

Ottiene il valore della proprietà "ShiftSize" (in percentuale).

```
double  ShiftSize() const
```

Valore di ritorno

Valore della proprietà "ShiftSize" del chart assegnato all'istanza della classe. Se non c'è tabella assegnata, restituisce EMPTY_VALUE.

ShiftSize (Metodo Set)

Imposta nuovo valore per la proprietà "Shift" (in percentuale).

```
bool  ShiftSize(
   double  shift_size      // valore della proprietà
   )
```

Parametri

shift_size

[in] Nuovo valore per la proprietà "ShiftSize" (in percentuale) .

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

AutoScroll (Metodo Get[Ottiene])

Ottiene il valore della proprietà "AutoScroll".

```
bool  AutoScroll() const
```

Valore di ritorno

Valore della proprietà "AutoScroll" del chart assegnato alla istanza della classe. Se non c'è chart assegnato, restituisce false.

AutoScroll (Metodo Set[Imposta])

Imposta nuovo valore per la proprietà "AutoScroll".

```
bool  AutoScroll(
   bool  autoscroll      // valore flag
   )
```

Parametri

autoscroll

[in] Nuovo valore per la proprietà "Autoscroll".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

Scale (Metodo Get)

Ottiene il valore della proprietà "Scala".

```
int  Scale() const
```

Valore di ritorno

Valore della proprietà "Scala" del chart assegnato all' istanza della classe. Se non c'è alcun chart assegnato, restituisce 0.

Scale (Metodo Set)

Imposta nuovo valore per la proprietà "Scale".

```
bool  Scale(
   int  scale      // valore della proprietà
   )
```

Parametri

scale

[in] Nuovo valore per la proprietà "Scale".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ScaleFix (Metodo Get)

Ottiene il valore della proprietà "ScaleFix" (scala chart fissa o no).

```
bool  ScaleFix() const
```

Valore di ritorno

Valore della proprietà "ScaleFix" del chart assegnato all'istanza della classe. Se non c'è chart assegnato, restituisce false.

ScaleFix (Metodo Set)

Imposta un nuovo valore per la proprietà "ScaleFix".

```
bool  ScaleFix(
   bool  scale_fix      // valore proprietà
   )
```

Parametri

scale_fix

[in] Nuovo valore per la proprietà "ScaleFix" .

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ScaleFix_11 (Metodo Get)

Ottiene il valore della proprietà "ScaleFix_11" (scala del chart è di 1:1, o no).

```
bool  ScaleFix_11() const
```

Valore di ritorno

Valore della proprietà "ScaleFix_11" del chart assegnato alla istanza della classe. Se non c'è alcun chart assegnato, restituisce false.

ScaleFix_11 (Metodo Set)

Imposta nuovo valore per la proprietà "ScaleFix_11".

```
bool  ScaleFix_11(
   string  scale_11      // valore della proprietà
   )
```

Parametri

scale_11

[in] Nuovo valore per la proprietà "ScaleFix_11".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

FixedMax (Metodo Get)

Ottiene il valore della proprietà "FixedMax" (prezzo massimo fissato).

```
double  FixedMax() const
```

Valore di ritorno

Valore della proprietà "FixedMax" del grafico assegnato alla istanza della classe. Se non c'è tabella assegnata, restituisce EMPTY_VALUE.

FixedMax (Metodo Set)

Imposta il nuovo valore per la proprietà "FixedMax".

```
bool  FixedMax(
   double  max      // massimo fissato
   )
```

Parametri

max

[in] Nuovo valore per la proprietà "FixedMax".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

FixedMin (Metodo Get)

Ottiene il valore della proprietà "FixedMin" (prezzo minimo fissato)

```
double  FixedMin() const
```

Valore di ritorno

Valore della proprietà "FixedMin" del grafico assegnato alla istanza della classe. Se non c'è tabella assegnata, restituisce EMPTY_VALUE.

FixedMin (Metodo Set)

Imposta nuovo valore per la proprietà "FixedMin".

```
bool  FixedMax(
   double  min      // minimo fissato
   )
```

Parametri

max

[in] Nuovo valore per la proprietà "FixedMin".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

PointsPerBar (Metodo Get)

Ottiene il valore della proprietà "PointsPerBar" (in punti per barra).

```
double  PointsPerBar() const
```

Valore di ritorno

Valore della proprietà di "PointsPerBar" del chart assegnato alla istanza della classe. Se non c'è chart assegnato, restituisce EMPTY_VALUE.

PointsPerBar (Metodo Set)

Imposta il nuovo valore per la proprietà "PointsPerBar".

```
bool  PointsPerBar(
   double  ppb      // scala
   )
```

Parametri

ppb

[in] Nuova scala (in points per barra).

Valore di ritorno

true - successo, false - non posso cambiare la scala.

ScalePPB (Metodo Get)

Ottiene il valore della proprietà "ScalePPB" (la scala è "punto per ogni barra" o no).

```
bool  ScalePPB() const
```

Valore di ritorno

Valore della proprietà "ScalePPB" del chart assegnato alla istanza della classe. Se non c'è alcun chart assegnato, restituisce false.

ScalePPB (Metodo Set)

Imposta nuovo valore per la proprietà "ScalePPB".

```
bool  ScalePPB(
   bool  scale_ppb      // valore flag
   )
```

Parametri

scale_ppb

[in] Nuovo valore della proprietà "ScalePPB".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowOHLC (Metodo Get)

Ottiene il valore della proprietà "ShowOHLC".

```
bool  ShowOHLC() const
```

Valore di ritorno

Valore della proprietà "ShowOHLC" del chart assegnato all'istanza della classe. Se non c'è chart assegnato, restituisce false.

ShowOHLC (Metodo Set)

Imposta nuovo valore per la proprietà "ShowOHLC".

```
bool  ShowOHLC(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore della proprietà "ShowOHLC".

Valore di ritorno

true - successo, false - non posso cambiare la proprietà.

ShowLineBid (Metodo Get)

Ottiene il valore della proprietà "ShowLineBid".

```
bool  ShowLineBid() const
```

Valore di ritorno

Valore della proprietà "ShowLineBid" del chart assegnato all'istanza della classe. Se non c'è chart assegnato, restituisce false.

ShowLineBid (Metodo Set)

Imposta nuovo valore per la proprietà "ShowLineBid".

```
bool  ShowLineBid(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore della proprietà "ShowLineBid".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowLineAsk (Metodo Get)

Ottiene il valore della proprietà "ShowLineAsk".

```
bool  ShowLineAsk() const
```

Valore di ritorno

Valore della proprietà "ShowLineAsk" del chart assegnato all'istanza della classe. Se non c'è chart assegnato, restituisce false.

ShowLineAsk (Metodo Set)

Imposta nuovo valore per la proprietà "ShowLineAsk".

```
bool  ShowLineAsk(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore della proprietà "ShowLineAsk".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowLastLine (Metodo Get)

Ottiene il valore della proprietà "ShowLastLine".

```
bool  ShowLastLine() const
```

Valore di ritorno

Valore della proprietà "ShowLastLine" del chart assegnato all' istanza della classe. Se non c'è chart assegnato, restituisce false.

ShowLastLine (Metodo Set)

Imposta nuovo valore per la proprietà "ShowLastLine".

```
bool  ShowLastLine(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore per la proprietà "ShowLastLine".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowPeriodSep (Metodo Get)

Ottiene il valore della proprietà "ShowPeriodSep" (mostra separatori periodo).

```
bool  ShowPeriodSep() const
```

Valore di ritorno

Valore della proprietà "ShowPeriodSep" del chart assegnato all'istanza della classe. Se non c'è alcun chart assegnato, restituisce false.

ShowPeriodSep (Metodo Set)

Imposta nuovo valore per la proprietà "ShowPeriodSep".

```
bool  ShowPeriodSep(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore della proprietà "ShowPeriodSep".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowGrid (Metodo Get)

Ottiene il valore della proprietà "ShowGrid".

```
bool  ShowGrid() const
```

Valore di ritorno

Valore della proprietà "ShowGrid" del chart assegnato all' istanza della classe. Se non c'è chart assegnato, restituisce false.

ShowGrid (Metodo Set)

Imposta nuovo valore per la proprietà "ShowGrid".

```
bool  ShowGrid(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore per la proprietà "ShowGrid".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowVolumes (Metodo Get)

Ottiene il valore della proprietà "ShowVolumes".

```
bool  ShowVolumes() const
```

Valore di ritorno

Valore della proprietà "ShowVolumes" del chart assegnato alla istanza della classe. Se non c'è chart assegnato, restituisce false.

ShowVolumes (Metodo Set)

Imposta nuovo valore per la proprietà "ShowVolumes".

```
bool  ShowVolumes(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore della proprietà "ShowVolumes".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowObjectDescr (Metodo Get)

Ottiene il valore della proprietà "ShowObjectDescr" (mostra descrizione per oggetti grafici).

```
bool  ShowObjectDescr() const
```

Valore di ritorno

Valore della proprietà "ShowObjectDescr" del chart assegnato all'istanza della classe. Se non c'è chart assegnato, restituisce false.

ShowObjectDescr (Metodo Set)

Imposta nuovo valore per la proprietà "ShowObjectDescr".

```
bool  ShowObjectDescr(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore della proprietà "ShowObjectDescr".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowDateScale

Imposta nuovo valore per la proprietà "ShowDateScale".

```
bool  ShowDateScale(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore per la proprietà "ShowDateScale".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ShowPriceScale

Imposta nuovo valore per la proprietà "ShowPriceScale".

```
bool  ShowPriceScale(
   bool  show      // valore della proprietà
   )
```

Parametri

show

[in] Nuovo valore per la proprietà "ShowOHLC".

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ColorBackground (Metodo Get)

Ottiene il valore della proprietà "ColorBackground" (colore di sfondo del chart).

```
color  ColorBackground() const
```

Valore di ritorno

Valore della proprietà "ColorBackground" del chart assegnata all' istanza della classe. Se non sono disponibili chart assegnati, restituisce CLR_NONE.

ColorBackground (Metodo Set)

Imposta nuovo valore per la proprietà "ColorBackground".

```
bool  ColorBackground(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore background.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorForeground (Metodo Get)

Ottiene il valore della proprietà "ColorForeground" (colore degli assi, scala e le stringhe OHLC del chart).

```
color  ColorForeground() const
```

Valore di ritorno

Valore della proprietà "ColorForeground" del chart assegnato all'istanza della classe. Se non sono disponibili chart assegnati, restituisce CLR_NONE.

ColorForeground (Metodo Set)

Imposta nuovo valore per la proprietà "ColorForeground" (per gli assi, la scala e stringa OHLC).

```
bool  ColorForeground(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore per gli assi, scala e stringa OHLC.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorGrid (Metodo Get)

Ottiene il valore della proprietà "ColorGrid" (colore della griglia).

```
color  ColorGrid() const
```

Valore di ritorno

Valore della proprietà "ColorGrid" del chart assegnato all' istanza della classe. Se non sono disponibili chart assegnati, restituisce CLR_NONE.

ColorGrid (Metodo Set)

Imposta nuovo valore per la proprietà "ColorGrid".

```
bool  ColorGrid(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore della griglia.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorBarUp (Get Method)

Ottiene il valore della proprietà "ColorBarUp" (colore per le barre bullish, loro ombra, e contorni del corpo della candela).

```
color  ColorBarUp() const
```

Valore di ritorno

Valore della proprietà "ColorBarUp" del chart assegnato alla istanza della classe. Se non sono disponibili chart assegnati, restituisce CLR_NONE.

ColorBarUp (Metodo Set)

Imposta nuovo valore per la proprietà "ColorBarUp".

```
bool  ColorBarUp(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore per le barre bullish, loro ombre, e contorni del corpo della candela.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorBarDown (Metodo Get)

Ottiene il valore della proprietà "ColorBarDown" (colore per le barre bearish, la loro ombra, e i contorni del corpo della candela).

```
color  ColorBarDown() const
```

Valore di ritorno

Valore della proprietà "ColorBarDown" del chart assegnato alla istanza della classe. Se non sono disponibili chart assegnati, restituisce CLR_NONE.

ColorBarDown (Metodo Set)

Imposta nuovo valore per la proprietà "ColorBarDown".

```
bool  ColorBarDown(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore per le barre bearish, loro ombre, e contorni del corpo della candela.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorCandleBull (Metodo Get)

Ottiene il valore della proprietà "ColorCandleBull" (colore del corpo della candela bullish).

```
color  ColorCandleBull() const
```

Valore di ritorno

Valore della proprietà "ColorCandleBull" del chart assegnato alla istanza della classe. Se non sono disponibili chart assegnati, restituisce CLR_NONE.

ColorCandleBull (Metodo Set)

Imposta nuovo valore per la proprietà "ColorCandleBull".

```
bool  ColorCandleBull(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore del corpo della candela bullish.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorCandleBear (Metodo Get)

Ottiene il valore della proprietà "ColorCandleBear" (colore del corpo della candela bearish).

```
color  ColorCandleBear() const
```

Valore di ritorno

Valore della proprietà "ColorCandleBear" del chart assegnato alla istanza della classe. Se non sono disponibili chart assegnati, restituisce CLR_NONE.

ColorCandleBear (Metodo Set)

Imposta il nuovo valore per la proprietà "ColorCandleBear".

```
bool  ColorCandleBear(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore del corpo della candela bearish.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorChartLine (Metodo Get)

Ottiene il valore della proprietà "ColorChartLine" (colore per la linea chart e candele Doji).

```
color  ColorChartLine() const
```

Valore di ritorno

Valore della "ColorChartLine" proprietà del chart assegnato all'istanza della classe. Se non ci sono chart assegnati, restituisce CLR_NONE.

ColorChartLine (Metodo Set)

Imposta nuovo valore per la proprietà "ColorChartLine".

```
bool  ColorChartLine(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore delle linee del chart e candele Doji.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorVolumes (Metodo Get)

Ottiene il valore della proprietà "ColorVolumes" (colore per volumi e livelli di posizioni aperte)

```
color  ColorVolumes() const
```

Valore di ritorno

Valore della proprietà "ColorVolumes" del chart assegnato all'istanza della classe. Se non ci sono chart assegnati, restituisce CLR_NONE.

ColorVolumes (Metodo Set)

Imposta nuovo valore per la proprietà "ColorVolumes".

```
bool  ColorVolumes(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore dei volumi e livelli della posizione aperta.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorLineBid (Metodo Get)

Ottiene il valore della proprietà "ColorLineBid" (colore della linea Bid)

```
color  ColorLineBid() const
```

Valore di ritorno

Valore della proprietà "ColorLineBid" del chart assegnato all' istanza della classe. Se non ci sono chart assegnati, restituisce CLR_NONE.

ColorLineBid (Metodo Set)

Imposta nuovo valore per la proprietà "ColorLineBid".

```
bool  ColorLineBid(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore per la linea Bid.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorLineAsk (Metodo Get)

Ottiene il valore della proprietà "ColorLineAsk" (colore della linea Ask).

```
color  ColorLineAsk() const
```

Valore di ritorno

Valore della proprietà "ColorLineAsk" del chart assegnato all' istanza della classe. Se non ci sono chart assegnati, restituisce CLR_NONE.

ColorLineAsk (Metodo Set)

Imposta nuovo valore per la proprietà "ColorLineAsk".

```
bool  ColorLineAsk(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore per la linea Ask.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorLineLast (Metodo Get)

Ottiene il valore della proprietà "ColorLineLast" (colore della linea di prezzo ultimo affare)

```
color  ColorLineLast() const
```

Valore di ritorno

Calore della proprietà "ColorLineLast" del chart assegnato all'istanza della classe. Se non ci sono chart assegnati, restituisce CLR_NONE.

ColorLineLast (Metodo Set)

Imposta nuovo valore per la proprietà "ColorLineLast".

```
bool  ColorLineLast(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore della linea di prezzo dell'ultimo affare.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

ColorStopLevels (Metodo Get)

Ottiene il valore della proprietà "ColorStopLevels" (colore dei livelli di SL e TP)

```
color  ColorStopLevels() const
```

Valore di ritorno

Valore della proprietà "ColorStopLevels" del chart assegnato all'istanza della classe. Se non ci sono chart assegnati, restituisce CLR_NONE.

ColorStopLevels (Metodo Set)

Imposta nuovo valore per la proprietà "ColorStopLevels".

```
bool  ColorStopLevels(
   color  new_color      // colore
   )
```

Parametri

new_color

[in] Nuovo colore dei livelli di prezzo Stop Loss e Take Profit.

Valore di ritorno

true - successo, false - non posso cambiare il colore.

VisibleBars

Ottiene il numero totale di barre del grafico visibili.

```
int  VisibleBars() const
```

Valore di ritorno

Ottiene il numero totale di barre visibili del chart assegnato alla istanza della classe. Se non c'è alcun chart assegnato, restituisce 0.

WindowsTotal

Ottiene il numero totale di finestre chart, comprese le sottofinestre chart indicatore.

```
int  WindowsTotal() const
```

Valore di ritorno

Numero complessivo di finestre, tra cui sottofinestre chart indicatore, assegnate alla istanza della classe. Se non c'è alcun chart assegnato, restituisce 0.

WindowIsVisible

Ottiene la flag visibilità della finestra secondaria del chart specificato.

```
bool  WindowIsVisible(
   int  num      // sottofinestra
   ) const
```

Parametri

num

[in] Numero sottofinestra (0 significa finestra principale).

Valore di ritorno

Restituisce la flag di visibilità della finestra chart secondaria specificata, assegnato all'istanza del chart. Se non c'è chart assegnato, restituisce false.

WindowHandle

Ottiene l'handle finestra del chart (HWND).

```
int  WindowHandle() const
```

Valore di ritorno

Handle della finestra del chart assegnato all'istanza chart. Se non sono disponibili chart assegnati, restituisce INVALID_HANDLE.

FirstVisibleBar

Ottiene il numero della prima barra visibile del chart.

```
int  FirstVisibleBar() const
```

Valore di ritorno

Numero della prima barra visibile del chart assegnato all'istanza della classe. Se non ci sono chart assegnati, restituisce -1.

WidthInBars

Ottiene larghezza del grafico, in barre.

```
int  WidthInBars() const
```

Valore di ritorno

Larghezza del chart in barre del chart assegnato alla istanza della classe. Se non c'è alcun chart assegnato, restituisce 0.

WidthInPixels

Ottiene larghezza del chart in pixel.

```
int  WidthInPixels() const
```

Valore di ritorno

Larghezza in pixel del chart assegnato all'istanza chart. Se non c'è alcun chart assegnato, restituisce 0.

HeightInPixels

Ottiene l'altezza della finestra in pixel.

```
int  HeightInPixels(
   int  num      // sottofinestra
   ) const
```

Parametri

num

[in] Numero sottofinestra controllato (0 significa finestra principale).

Valore di ritorno

Altezza della finestra in pixel del chart assegnato alla istanza della classe. Se non c'è alcun chart assegnato, restituisce 0.

PriceMin

Ottiene il minimo prezzo della finestra.

```
double  PriceMin(
   int  num      // sottofinestra
   ) const
```

Parametri

num

[in] Numero sottofinestra (0 significa finestra principale).

Valore di ritorno

Valore del prezzo minimo della finestra del chart assegnata alla istanza della classe. Se non c'è un chart assegnato, restituisce EMPTY_VALUE.

PriceMax

Ottiene il massimo prezzo della finestra.

```
double  PriceMax(
   int  num      // sottofinestra
   ) const
```

Parametri

num

[in] Numero sottofinestra (0 significa finestra principale).

Valore di ritorno

Valore del prezzo massimo della finestra del chart assegnato alla istanza della classe. Se non c'è un chart assegnato, restituisce EMPTY_VALUE.

Attach

Assegna il chart corrente all'istanza della classe.

```
void  Attach()
```

Attach

Assegna il chart specificato all'istanza della classe.

```
void  Attach(
   long  chart      // identificatore chart
   )
```

Parametri

chart

[in] Identificatore del grafico assegnato.

FirstChart

Assegna il primo chart del terminale client all'istanza della classe.

```
void  FirstChart()
```

NextChart

Assegna il prossimo (successivo a quello già assegnato) all'istanza della classe.

```
void  NextChart()
```

Open

Apre il chart con i parametri specificati e lo assegna alla istanza della classe.

```
long  Open(
   const string      symbol_name,      // simbolo
   ENUM_TIMEFRAMES   timeframe         // periodo
   )
```

Parametri

symbol_name

[in] Simbolo del Chart. NULL indica il simbolo del chart corrente (a cui è collegato un expert).

timeframe

[in] Chart timeframe (dall'enumerazione ENUM_TIMEFRAMES). 0 significa il timeframe corrente.

Valore di ritorno

identificatore chart.

Detach

Stacca il chart dalla istanza della classe.

```
void  Detach()
```

Close

Chiude il grafico assegnato alla istanza della classe.

```
void  Close()
```

BringToTop

Mostra il chart sopra ad altri charts.

```
bool  BringToTop() const
```

Valore di ritorno

true - successo, false - errore.

EventObjectCreate

Imposta un flag per inviare notifiche di un evento di una creazione oggetto grafico.

```
bool  EventObjectCreate(
   bool  flag      // flag
   )
```

Parametri

flag

[in] Nuovo valore di un flag per inviare notifiche di un evento di creazione un oggetto grafico.

Valore di ritorno

true - successo, falso - non si può cambiare la flag.

EventObjectDelete

Imposta un flag per l'invio della notifica di eventi di cancellazione di oggetto grafico.

```
bool  EventObjectDelete(
   bool  flag      // flag
   )
```

Parametri

flag

[in] Nuovo valore di un flag per l'invio della notifica di eventi di eliminazione di oggetto grafico.

Valore di ritorno

true - successo, falso - non si può cambiare la flag.

IndicatorAdd

Aggiunge un indicatore con l'handle specificato in una finestra chart specificata.

```
bool  IndicatorAdd(
   int   sub_win         // numero della sottofinestra
   int   handle          // handle dell'indicatore
   );
```

Parametri

sub_win

[in] Il numero della sottofinestra chart. 0 significa la finestra del chart principale. se è specificato il numero di una finestra non esistente, verrà creata una nuova finestra.

handle

[in] L' handle dell' indicatore.

Valore di ritorno

La funzione restituisce true in caso di successo, altrimenti restituisce false. Per ottenere informazioni sull' errore, chiamare la funzione GetLastError().

Vedi anche

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

IndicatorDelete

Rimuove un indicatore con il nome specificato dalla finestra chart specificata.

```
bool  IndicatorDelete(
   int            sub_win       // numero della sottofinestra
   const string   name          // nome breve dell'indicatore
   );
```

Parametri

sub_win

[in] Numero della sottofinestra chart. 0 indica la sottofinestra chart principale.

const name

[in] Il nome breve dell'indicatore che è impostato nella proprietà INDICATOR_SHORTNAME con la funzione IndicatorSetString(). Per ottenere il nome breve di un indicatore, utilizzare la funzione IndicatorName().

Valore di ritorno

Restituisce true in caso di successo dell'eliminazione dell'indicatore. In caso contrario, restituisce false. Per ottenere i dettagli dell'errore, usare la funzione GetLastError().

Nota

Se sono presenti due indicatori con nomi brevi identici nella sottofinestra chart, il primo in fila verrà eliminato.

Se altri indicatori su questo chart si basano sui valori dell'indicatore che viene cancellato, tali indicatori verranno eliminati .

Non confondere il nome breve indicatore e il nome del file specificato durante la creazione di un indicatore con le funzioni iCustom() ed IndicatorCreate(). Se il nome breve di un indicatore non è impostato in modo esplicito, verrà specificato il nome del file che contiene il codice sorgente dell'indicatore durante la compilazione.

La cancellazione di un indicatore da un chart non significa che la sua parte di calcolo sarà cancellata dalla memoria del terminale. Per rilasciare l'handle dell indicatore, utilizzare la funzione IndicatorRelease().

Il nome breve dell'indicatore dovrebbe essere formato correttamente. Sarà scritto nella proprietà INDICATOR_SHORTNAME utilizzando la funzione IndicatorSetString(). Si raccomanda che il nome breve deve contenere valori di tutti i parametri di input dell'indicatore, perché l'indicatore da eliminare dal chart dalla funzione IndicatorDelete() è identificato dal nome breve.

Vedi anche

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

IndicatorsTotal

Restituisce il numero di tutti gli indicatori applicati alla finestra chart specificata.

```
int  IndicatorsTotal(
   long    chart_id,  // identificatore chart
   int     sub_win    // numero della sottofinestra
   );
```

Parametri

chart_id

[in] Identificatore del Chart. 0 denota il grafico principale.

sub_win

[in] Numero della sottofinestra chart. 0 indica la finestra grafico principale.

Valore di ritorno

Il numero di indicatori nella finestra chart specificata. Per ottenere i dettagli dell'errore, usare la funzione GetLastError().

Nota

La funzione permette di andare a cercare tra tutti gli indicatori collegati al chart. Il numero di tutte le finestre del chart può essere ottenuto dalla struttura CHART_WINDOWS_TOTAL utilizzando la funzione GetInteger().

Vedi anche

IndicatorAdd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

IndicatorName

Restituisce il nome breve dell'indicatore per l'indice nella lista degli indicatori della finestra chart specificata.

```
string  IndicatorName(
   int    sub_win      // numero della sottofinestra
   int    index        // indice dell'indicatore nell'elenco degli indicatori aggiunto a
   );
```

Parametri

sub_win

[in] Numero della sottofinestra chart. 0 indica la finestra grafico principale.

index

[in] Indice dell'indicatore nell'elenco degli indicatori. La numerazione degli indicatori inizia con zero, cioè il primo indicatore nella lista ha indice 0. Per ottenere il numero di indicatori nella lista, utilizzare la funzione IndicatorsTotal().

Valore di ritorno

Il nome breve dell'indicatore che si trova nella proprietà INDICATOR_SHORTNAME con la funzione IndicatorSetString(). Per ottenere i dettagli dell'errore, usare la funzione GetLastError().

Nota

Non confondere il nome breve indicatore e il nome del file specificato durante la creazione di un indicatore con le funzioni iCustom() ed IndicatorCreate(). Se il nome breve di un indicatore non è impostato in modo esplicito, verrà specificato il nome del file che contiene il codice sorgente dell'indicatore durante la compilazione.

Il nome breve dell'indicatore dovrebbe essere formato correttamente. Sarà scritto nella proprietà INDICATOR_SHORTNAME utilizzando la funzione IndicatorSetString(). Si raccomanda che il nome breve deve contenere valori di tutti i parametri di input dell'indicatore, perché l'indicatore da eliminare dal chart dalla funzione IndicatorDelete() è identificato dal nome breve.

Vedi anche

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

Navigate

Slitta il grafico.

```
bool  Navigate(
   ENUM_CHART_POSITION  position,     // posizione
   int                  shift=0       // lo shift
   )
```

Parametri

position

[in] Posizione del chart (dall'enumerazione ENUM_CHART_POSITION), rispetto alla quale viene eseguito lo slittamento.

shift=0

[in] Numero di barre da slittare.

Valore di ritorno

true - successful, false - non si può slittare il chart.

Symbol

Ottiene il nome del simbolo del chart.

```
string  Symbol() const
```

Valore di ritorno

Nome del simbolo del chart, assegnato all'istanza della classe. Se non c'è alcun chart assegnato, esso restituisce "".

Period

Ottiene il periodo del chart.

```
ENUM_TIMEFRAMES  Period() const
```

Valore di ritorno

Periodo del chart (da ENUM_TIMEFRAMES) assegnato alla istanza della classe. Se non vi è alcun chart assegnato, restituisce 0.

Redraw

Ridisegna chart assegnato all'istanza della classe.

```
void  Redraw()
```

GetInteger

La funzione restituisce il valore della corrispondente proprietà chart. La proprietà chart dev' essere di tipo integer. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // identificatore proprietà
   int                          sub_window=0      // numero sottofinestra
   ) const
```

2. In caso di successo, mette il valore della proprietà alla variabile specificata di tipo integer, passata per riferimento come ultimo parametro.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,        // identificatore proprietà
   int                          sub_window,     // numero sottofinestra
   long&                        value           // link alla variabile
   ) const
```

Parametri

prop_id

[in] identificatore della proprietà (enumerazione ENUM_CHART_PROPERTY_INTEGER).

sub_window

[in] Numero sottofinestra Chart.

value

[in] Link della variabile che riceve il valore della proprietà richiesta.

Valore di ritorno

Valore della proprietà del chart assegnato alla istanza della classe. Se non c'è alcun chart assegnato, restituisce -1.

Per la seconda variante, la funzione restituisce true, se questa proprietà è mantenuta ed il valore è stato posto nel valore della variabile, altrimenti restituisce false. Per saperne di più riguardo l'errore, chiamare GetLastError().

SetInteger

Imposta nuovo valore per la proprietà di tipo integer.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // identificatore propriet<
   long                         value         // valore
   )
```

Parametri

prop_id

[in] identificatore della proprietà chart (dall'enumerazione ENUM_CHART_PROPERTY_INTEGER).

value

[in] Nuovo valore della proprietà.

Valore di ritorno

true - successo, false - non posso cambiare la proprietà integer.

GetDouble

La funzione restituisce il valore della corrispondente proprietà chart. La proprietà oggetto deve essere di tipo double. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // identificatore proprietà
   int                         sub_window=0     // numero sottofinestra
   ) const
```

2. In caso di successo, mette il valore della proprietà nella variabile specificata di tipo double, passata per riferimento come ultimo parametro.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // identificatore proprietà
   int                         sub_window,     // numero sottofinestra
   double&                     value           // link alla variabile
   ) const
```

Parametri

prop_id

[in] Identificatore della proprietà del Chart (dall'enumerazione ENUM_CHART_PROPERTY_DOUBLE).

sub_window

[in] Numero sottofinestra Chart.

value

[in] Variabile di tipo double, che ha ricevuto il valore della proprietà richiesta.

Valore di ritorno

Valore della proprietà del grafico assegnato alla istanza della classe. Se non c'è alcuna tabella assegnata, restituisce EMPTY_VALUE.

Per la seconda variante della funzione, restituisce true se viene ricevuto il valore della proprietà, in caso contrario restituisce false. Per saperne di più riguardo l'errore, chiamare GetLastError().

SetDouble

Imposta nuovo valore per la proprietà del chart, di tipo double.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // identificatore proprietà
   double                      value         // nuovo valore
   )
```

Parametri

prop_id

[in] Identificatore della proprietà del Chart (dall'enumerazione ENUM_CHART_PROPERTY_DOUBLE).

value

[in] Nuovo valore per la proprietà.

Valore di ritorno

true - successo, false - non posso cambiare il valore della proprietà double.

GetString

La funzione restituisce il valore della corrispondente proprietà chart. La proprietà del grafico dovrebbe essere di tipo stringa. Ci sono due varianti della funzione.

1. Restituisce immediatamente il valore della proprietà.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // identificatore proprietà
   ) const
```

2. In caso di successo, mette il valore della proprietà nella variabile specificata di tipo string, passata per riferimento come ultimo parametro.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // identificatore proprietà
   string&                     value        // link alla variabile
   ) const
```

Parametri

prop_id

[in] Identificatore della proprietà Chart (dall'enumerazione ENUM_CHART_PROPERTY_STRING).

value

[in] Link della variabile che riceve il valore della proprietà richiesta.

Valore di ritorno

Valore della proprietà chart assegnata all' istanza della classe. Se non c'è alcun chart assegnato, esso restituisce "".

Per la seconda variante, la funzione restituisce true, se questa proprietà è mantenuta ed il valore è stato posto nel valore della variabile, altrimenti restituisce false. Per saperne di più riguardo l'errore, chiamare GetLastError().

SetString

Imposta nuovo valore per la proprietà chart, di tipo stringa.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // identificatore proprietà
   string                      value         // valore
   )
```

Parametri

prop_id

[in] Identificatore della proprietà Chart (dall'enumerazione ENUM_CHART_PROPERTY_STRING).

value

[in] Nuovo valore per la proprietà.

Valore di ritorno

true - successo, false - non posso cambiare la proprietà string.

SetSymbolPeriod

Cambia il simbolo ed il periodo del chart assegnato all' istanza della classe.

```
bool  SetSymbolPeriod(
   const string      symbol_name,      // simbolo
   ENUM_TIMEFRAMES   timeframe         // periodo
   )
```

Parametri

symbol_name

[in] Nuovo simbolo chart. NULL indica il simbolo del chart corrente (a cui è collegato un expert).

timeframe

[in] Nuovo chart timeframe (dall'enumerazione ENUM_TIMEFRAMES). 0 significa il timeframe corrente del chart.

Valore di ritorno

true - successo, false - non si può cambiare la proprietà.

ApplyTemplate

Applica il template specificato al grafico.

```
bool  ApplyTemplate(
   const string  filename      // template
   )
```

Parametri

filename

[in] Nome file del template.

Valore di ritorno

true - successo, false - non posso applicare il template.

ScreenShot

Crea uno screenshot del chart specificato nel suo stato attuale in formato .gif.

```
bool  ScreenShot(
   string           filename,                  // nome file
   int              width,                     // larghezza
   int              height,                    // altezza
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // tipo di allineamento
   ) const
```

Parametri

filename

[in] Nome file per lo screenshot.

width

[in] Larghezza screenshot in pixels.

height

[in] Altezza screenshot in pixels.

align_mode=ALIGN_RIGHT

[in] Modalità allineamento, se lo screenshot è stretto.

Valore di ritorno

true - successo, false - errore.

WindowOnDropped

Ottiene il numero della sottofinestra chart corrispondente al punto di rilascio dell'oggetto (expert o script)

```
int  WindowOnDropped() const
```

Valore di ritorno

Numero sottofinestra chart del punto di rilascio dell'oggetto. 0 significa finestra del chart principale.

PriceOnDropped

Ottiene la coordinata prezzo corrispondente al rilascio dell'oggetto (expert o script) .

```
double  PriceOnDropped() const
```

Valore di ritorno

Coordinate Prezzo del punto di rilascio dell'oggetto.

TimeOnDropped

Ottiene le coordinate tempo corrispondenti al punto di rilascio dell'oggetto (expert o script)

```
datetime  TimeOnDropped() const
```

Valore di ritorno

Coordinate Termpo del punto di rilascio dell'oggetto.

XOnDropped

Ottiene le coordinate X corrispondenti al punto di rilascio dell'oggetto (expert o script)

```
int  XOnDropped() const
```

Valore di ritorno

Coordinate X del punto di rilascio dell'oggetto.

YOnDropped

Ottiene le coordinate Y corrispondenti al punto di rilascio dell'oggetto (expert o script).

```
int  YOnDropped() const
```

Valore di ritorno

Coordinate Y del punto di rilascio dell'oggetto.

Save

Salva parametri degli oggetti in un file.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] l'handle del file binario già aperto dalla funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Load

Caricha i parametri dell'oggetto da file.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parametri

file_handle

[in] l'handle del file binario già aperto dalla funzione FileOpen(...).

Valore di ritorno

true - successo, false - errore.

Type

Restituisce il tipo d'identificatore.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di identificatore (0x1111 per CChart).

Graphics

La libreria grafica contiene classi e funzioni globali per un rapido disegno di grafici personalizzati. La libreria fornisce convenienti soluzioni già pronte per la costruzione di assi, curve, così come le modalità di accesso rapido alle modifica delle proprietà comuni di un chart personalizzato.

Per iniziare a lavorare con la libreria, leggi semplicemente l'articolo Visualizza questo! Libreria grafica MQL5 simile al 'plot' del linguaggio R.

La libreria grafica è posizionata nella cartella Include\Graphics della directory di lavoro del terminale.

Classe	Descrizione
CAxis	Classe per lavorare con gli assi cartesiani
ColorGenerator	Classe che specifica la combinazione di colori di default
CCurve	Classe per lavorare con le curve
CGraphic	Classe base per la creazione di charts personalizzati

GraphPlot

Funzioni per la rapida tracciatura della curva.

Versione per il disegno di una singola curva utilizzando le coordinate Y.

```
string  GraphPlot(
   const double      &y[],                  // Coordinate Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo di curva
   )
```

Nota

Indici di array Y sono utilizzati come coordinate X per la curva.

Versione per il disegno di una singola curva utilizzando le coordinate X ed Y.

```
string  GraphPlot(
   const double      &x[],                  // Coordinate X
   const double      &y[],                  // Coordinate Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo di curva
   )
```

Versione per il disegno di due curve utilizzando le coordinate X ed Y.

```
string  GraphPlot(
   const double      &x1[],                 // Coordinate X
   const double      &y1[],                 // Coordinate Y
   const double      &x2[],                 // Coordinate X
   const double      &y2[],                 // Coordinate Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo di curva
   )
```

Versione per il disegno di tre curve utilizzando coordinate X e Y

```
string  GraphPlot(
   const double      &x1[],                 // Coordinate X
   const double      &y1[],                 // Coordinate Y
   const double      &x2[],                 // Coordinate X
   const double      &y2[],                 // Coordinate Y
   const double      &x3[],                 // Coordinate X
   const double      &y3[],                 // Coordinate Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo di curva
   )
```

Versione per il disegno di una curva usando i punti delle coordinate CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points[],            // coordinate della curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo di curva
   )
```

Versione per il disegno di due curve usando i punti delle coordinate CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordinate della curva
   const CPoint2D   &points2[],           // coordinate della curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo di curva
   )
```

Versione per il disegno di tre curve usando i punti delle coordinate CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordinate della curva
   const CPoint2D   &points2[],           // coordinate della curva
   const CPoint2D   &points3[],           // coordinate della curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo di curva
   )
```

Versione per il disegno di una curva utilizzando il puntatore a CurveFunction

```
string  GraphPlot(
   CurveFunction    function,             // puntatore alla funzione
   const double     from,                 // valore iniziale dell'argomento
   const double     to,                   // valore finale dell'argomento
   const double     step,                 // incremento per argomento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo di curva
   )
```

Versione per il disegno di due curve utilizzando i puntatori alle funzioni CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,            // puntatore alla funzione
   CurveFunction    function2,            // puntatore alla funzione
   const double     from,                 // valore iniziale dell'argomento
   const double     to,                   // valore finale dell'argomento
   const double     step,                 // incremento per argomento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo di curva
   )
```

Versione per il disegno di tre curve utilizzando i puntatori alle funzioni CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,            // puntatore alla funzione
   CurveFunction    function2,            // puntatore alla funzione
   CurveFunction    function3,            // puntatore alla funzione
   const double     from,                 // valore iniziale dell'argomento
   const double     to,                   // valore finale dell'argomento
   const double     step,                 // incremento per argomento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo di curva
   )
```

Parametri

&x[]

[in] Coordinate X.

&y[]

[in] Coordinate Y.

&x1[]

[in] Coordinate X per la prima curva.

&y1[]

[in] Coordinate Y per la prima curva.

&x2[]

[in] Coordinate X per la seconda curva.

&y2[]

[in] Coordinate Y per la seconda curva.

&x3[]

[in] Coordinate X per la terza curva.

&y3[]

[in] Coordinate Y per la terza curva.

&points[]

[in] Coordinate dei punti della curva.

&points1[]

[in] Coordinate dei primi punti della curva.

&points2[]

[in] Coordinate dei secondi punti della curva.

&points3[]

[in] Coordinate dei terzi punti della curva.

function

[in] Puntatore alla funzione CurveFunction.

function1

[in] Puntatore alla prima funzione.

function2

[in] Puntatore alla seconda funzione.

function3

[in] Puntatore alla terza funzione.

from

[in] Corrisponde alla prima coordinata X.

to

[in] corrisponde all'ultima coordinata X.

step

[in] Il parametro per il calcolo delle coordinate X.

type=CURVE_POINTS

[in] Tipo di curva.

Valore di ritorno

Nome di una risorsa grafica.

CAxis

CAxis è una classe di libreria grafica ausiliaria per lavorare con le coordinate degli assi.

Descrizione

La classe CAxis riceve e memorizza vari parametri delle coordinate degli assi. La classe implementa la capacità di scalare automaticamente gli assi delle coordinate in modo dinamico.

Dichiarazione

```
class CAxis
```

Title

```
#include <Graphics\Axis.mqh>
```

Metodi della Classe

Metodo	Descrizione
AutoScale	Ottiene/Imposta la flag di scala automatica
Min	Ottiene/Imposta il valore minimo dell'asse
Max	Ottiene/imposta il valore massimo dell'asse
Step	Ottiene il valore di step per asse
Name	Ottiene/imposta il nome dell'asse
Color	Ottine/imposta il colore dell'asse
ValuesSize	Ottiene/imposta la grandezza dei numeri degli assi
ValuesWidth	Ottiene/imposta la lunghezza massima visualizzata dei numeri degli assi
ValuesFormat	Ottiene/imposta il formato dei numeri degli assi
ValuesDateTimeMode	Ottiene il formato di conversione di una data in una stringa.
ValuesFunctionFormat	Ottieni il puntatore alla funzione che definisce il formato di visualizzazione dei valori sull'asse.
ValuesFunctionFormatCBData	Ottiene il puntatore all'oggetto che può contenere dati aggiuntivi sulla conversione dei valori degli assi.
NameSize	Ottiene/imposta la dimensione del carattere del nome dell'asse
ZeroLever	Ottiene/imposta il valore "zero lever"
DefaultStep	Ottiene/imposta il valore iniziale del passo per asse

Metodo	Descrizione
MaxLabels	Ottiene/imposta la quantità massima di numeri sull'asse
MinGrace	Ottiene/imposta il valore "tolleranza" per l'asse minimo
MaxGrace	Ottiene/imposta il valore "tolleranza" per l'asse massimo
SelectAxisScale	Auto scala l'asse.

AutoScale (Metoto Get)

Restituisce la flag che definisce la necessità di auto-scaltatura.

```
bool  AutoScale()
```

Valore di ritorno

Il valore del flag.

Nota

true — fa l' auto-scaling.

false — non fa l' auto-scaling.

AutoScale (Metoto Set)

Imposta la flag che definisce la necessità di auto-scaltatura.

```
void  AutoScale(
   const bool  auto      // valore flag
   )
```

Parametri

auto

[in]

Nota

true — fa l' auto-scaling.

false — non fa l' auto-scaling.

Min (Metodo Get)

Restituisce il valore minimo dell'asse.

```
double  Min()
```

Valore di ritorno

Valore minimo dell'asse.

Min (Metodo Set)

Imposta il valore minimo dell'asse.

```
void  Min(
   const double  min      // valore minimo
   )
```

Parametri

min

[in] Valore minimo.

Max (Metodo Get)

Restituisce il valore massimo dell'asse.

```
double  Max()
```

Valore di ritorno

Valore massimo dell'asse.

Max (Metodo Set)

Imposta il valore massimo dell'asse.

```
void  Max(
   const double  max      // valore massimo
   )
```

Parametri

max

[in] Massimo valore dell'asse.

Step (Metodo Get)

Restituisce il valore distep per asse.

```
double  Step()
```

Valore di ritorno

Valore Step.

Name (Metodo Get)

Restituisce il nome dell'asse.

```
string  Name()
```

Valore di ritorno

Nome dell'asse.

Name (Metodo Set)

Imposta il nome dell'asse.

```
void  Name(
   const string  name      // nome dell'asse
   )
```

Parametri

name

[in] Nome dell'asse.

Color (Metodo Get)

Restituisce il colore dell'asse.

```
color  Color()
```

Valore di ritorno

Colore dell'asse.

Color (Metodo Set)

Imposta il colore dell'asse.

```
void  Color(
   const color  clr      // colore dell'asse
   )
```

Parametri

clr

[in] Colore dell'asse.

ValuesSize (Metodo Get)

Restituisce la dimensione dei numeri dell'asse.

```
int  ValuesSize()
```

Valore di ritorno

Dimensioni dei numeri dell' asse.

ValuesSize (Metodo Set)

Imposta la dimensione del numero dell'asse.

```
void  ValuesSize(
   const int  size      // grandezza dei numeri dell'asse
   )
```

Parametri

size

[in] Dimensione dei numeri dell'asse

ValuesWidth (Metodo Get)

Restituisce la lunghezza massima consentita in pixel per la visualizzazione dei numeri dell'asse.

```
int  ValuesWidth()
```

Valore di ritorno

Lunghezza dei numeri degli assi in pixel.

Nota

Se la lunghezza in pixel per un numero specificato supera la lunghezza massima consentita del display, è troncata e termina in punti.

ValuesWidth (Metodo Set)

Imposta la lunghezza massima consentita in pixel per la visualizzazione dei numeri degli assi.

```
void  ValuesWidth(
   const int  width      // massima lunghezza consentita in pixel
   )
```

Parametri

width

[in] Massima lunghezza consentita dei numeri degli assi.

Nota

Se la lunghezza in pixel per un numero specificato supera la lunghezza massima consentita del display, è troncata e termina in punti.

ValuesFormat (Metodo Get)

Restituisce il formato dei numeri degli assi.

```
string  ValuesFormat()
```

Valore di ritorno

Formato del numero.

ValuesFormat (Metodo Set)

Imposta il formato dei numeri degli assi.

```
void  ValuesFormat(
   const string  format      // formato del numero degli assi
   )
```

Parametri

format

[in] Formato del numero degli assi.

ValuesDateTimeMode (Metodo Get)

Ottiene il formato di conversione di una data in una stringa.

```
int  ValuesDateTimeMode()
```

Return Value

Formato di conversione di una data in una stringa.

ValuesDateTimeMode (Metodod Set)

Impostare il formato di conversione di una data in una stringa.

```
void  ValuesDateTimeMode(
   const int  mode      // formato di conversione di una data in una stringa
   )
```

Parametri

mode

[in] Formato di conversione.

Nota

Scopri di più sui formati di conversione di una data in una stringa nella descrizione della funzione TimeToString() .

ValuesFunctionFormat (Metodo Get)

Ottieni il puntatore alla funzione che definisce il formato di visualizzazione dei valori sull'asse.

```
DoubleToStringFunction  ValuesFunctionFormat()
```

Return Value

Puntatore alla funzione che definisce il formato di visualizzazione dei valori sull'asse.

ValuesFunctionFormat (Metodo Set)

Impostare il puntatore alla funzione che definisce il formato di visualizzazione dei valori sull'asse.

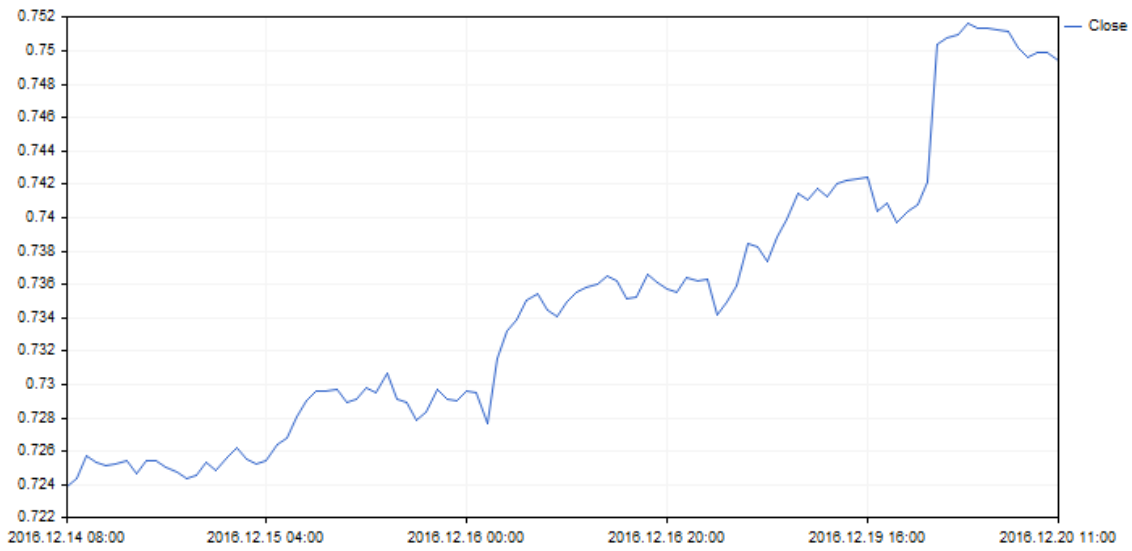
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // funzione per convertire valori numerici in sti
   )
```

Parametri

func

[in] Funzione personalizzata per la conversione di valori numerici in una stringa.

Esempio:



Il formato di visualizzazione dei valori dell'asse X è stato modificato utilizzando il seguente codice:

```
//+------------------------------------------------------------------+
//|                                              DateAxisGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//--- array per memorizzare i valori
double arrX[];
double arrY[];
//+------------------------------------------------------------------+
//| Funzione Custom per creare i valori sull' asse X                 |
//+------------------------------------------------------------------+
string TimeFormat(double x,void *cbdata)
  {
   return(TimeToString((datetime)arrX[ArraySize(arrX)-(int)x-1]));
  }
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[];
   CopyRates(Symbol(),Period(),0,100,rates);
   ArraySetAsSeries(rates,true);
   int size=ArraySize(rates);
   ArrayResize(arrX,size);
   ArrayResize(arrY,size);
   for(int i=0; i<size;++i)
     {
      arrX[i]=(double)rates[i].time;
      arrY[i]=rates[i].close;
     }
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"DateAxisGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"DateAxisGraphic");
     }
//--- crea curva
   CCurve *curve=graphic.CurveAdd(arrY,CURVE_LINES);
//--- ottiene l'asse delle X
   CAxis *xAxis=graphic.XAxis();
//--- imposta proprietà asse delle X
   xAxis.AutoScale(false);
   xAxis.Type(AXIS_TYPE_CUSTOM);
   xAxis.ValuesFunctionFormat(TimeFormat);
   xAxis.DefaultStep(20.0);
//--- disegna
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

ValuesFunctionFormatCBData (Metodo Get)

Ottiene il puntatore all'oggetto che può contenere dati aggiuntivi sulla conversione dei valori degli assi.

```
void*  ValuesFunctionFormatCBData()
```

Return Value

Il puntatore all'oggetto che può contenere dati aggiuntivi sulla conversione dei valori degli assi .

ValuesFunctionFormatCBData (Metodo Set)

Imposta il puntatore nell'oggetto classe che può contenere dati aggiuntivi sulla conversione dei valori degli assi.

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // puntatore all'ggetto della classe
   )
```

Parametri

cbdata

[in] Puntatore a qualsiasi oggetto della classe contenente dati aggiuntivi sulla conversione dei valori degli assi

NameSize (Metodo Get)

Restituisce la grandezza del carattere del nome dell'asse.

```
int  NameSize()
```

Valore di ritorno

Dimensione del carattere del nome dell'asse.

NameSize (Metodo Set)

Imposta la dimensione del carattere del nome dell'asse.

```
void  NameSize(
   const int  size      // grandezza del font del nome dell'asse
   )
```

Parametri

size

[in] Dimensione del carattere del nome dell'asse.

ZeroLever (Metodo Get)

Retuns il valore "zero lever".

```
double  ZeroLever()
```

Valore di ritorno

"Zero lever".

Nota

Il valore viene utilizzato per definire quando l'intervallo di scala dell'asse dovrebbe essere ampliato per includere un valore pari a zero.

ZeroLever (Metodo Set)

Imposta il valore "zero lever".

```
void  ZeroLever(
   const double  value      // valore "zero lever"
   )
```

Parametri

value

[in] Valore "Zero lever".

Nota

Il valore viene utilizzato per definire quando l'intervallo di scala dell'asse dovrebbe essere ampliato per includere un valore pari a zero.

DefaultStep (Metodo Get)

Restituisce il valore dello step iniziale per asse

```
double  DefaultStep()
```

Valore di ritorno

Step per asse.

DefaultStep (Metoto Set)

Imposta il valore step iniziale per asse

```
void  DefaultStep(
   const double  value      // step per asse
   )
```

Parametri

value

[in] Valore iniziale dello step per asse.

MaxLabels (Metodo Get)

Restituisce la quantità massima consentita di numeri visualizzati sull'asse.

```
double  MaxLabels()
```

Valore di ritorno

Ammontare massimo di numeri sull'asse.

MaxLabels (Metodo Set)

Imposta la quantità massima consentita di numeri visualizzati sull'asse.

```
void  MaxLabels(
   const double  value      // numero massimo
   )
```

Parametri

value

[in] Massimo ammontare consentito di numeri visualizzati sull'asse

MinGrace (Metodo Get)

Restituisce la "tolleranza" applicata al minimo dell'asse

```
double  MinGrace()
```

Valore di ritorno

Valore della "tolleranza" per il minimo dell'asse

Nota

Questo valore è espresso come parte della lunghezza assiale complessiva. Ad esempio, supponiamo che i valori degli assi si trovano da 4.0 a 16,0, pertanto la sua lunghezza è di 12.0. Se MinGrace è uguale a 0.1, allora il 10% della lunghezza dell'asse (o 1.2) viene sottratto dal valore minimo. Di conseguenza, l'asse copre l'intervallo da 2.8 a 16.0.

MinGrace (Metodo Set)

Imposta la "tolleranza" applicata al minimo dell'asse

```
void  MinGrace(
   const double  value      // valore "tolleranza"
   )
```

Parametri

value

[in] "Tolleranza" applicata al minimo dell'asse.

Nota

Questo valore è espresso come parte della lunghezza assiale complessiva. Ad esempio, supponiamo che i valori degli assi si trovano da 4.0 a 16,0, pertanto la sua lunghezza è di 12.0. Se MinGrace è uguale a 0.1, allora il 10% della lunghezza dell'asse (o 1.2) viene sottratto dal valore minimo. Di conseguenza, l'asse copre l'intervallo da 2.8 a 16.0.

MaxGrace (Metodo Get)

Restituisce la "tolleranza" applicata al massimo dell'asse

```
double  MaxGrace()
```

Valore di ritorno

Valore della "tolleranza" per il massimo dell'asse

Nota

Questo valore è espresso come parte della lunghezza assiale complessiva. Ad esempio, supponiamo che i valori degli assi si trovano da 4.0 a 16,0, pertanto la sua lunghezza è di 12.0. Se MaxGrace è uguale a 0.1, allora il 10% della lunghezza dell'asse (o 1.2) viene aggiunta al valore massimo. Di conseguenza, l'asse copre l'intervallo da 4.0 a 17.2.

MaxGrace (Metodo Set)

Imposta la "tolleranza" applicata al massimo dell'asse

```
void  MaxGrace(
   const double  value      // valore "tolleranza"
   )
```

Parametri

value

[in] Valore "tolleranza" applicato al massimo dell'asse.

Nota

Questo valore è espresso come parte della lunghezza assiale complessiva. Ad esempio, supponiamo che i valori degli assi si trovano da 4.0 a 16,0, pertanto la sua lunghezza è di 12.0. Se MinGrace è uguale a 0.1, allora il 10% della lunghezza dell'asse (o 1.2) viene sottratto dal valore minimo. Di conseguenza, l'asse copre l'intervallo da 2.8 a 16.0.

SelectAxisScale

Auto scala l'asse.

```
void  SelectAxisScale()
```

CColorGenerator

La Classe CColorGenerator è una classe ausiliaria della libreria grafica per lavorare con la tavolozza dei colori.

Descrizione

La classe CColorGenerator contiene la tavolozza dei colori iniziali utilizzati per le curve per default (se un colore non è specificato dall'utente).

Se tutti i colori dalla tavolozza iniziale vengono utilizzati già, nuovi colori vengono generati automaticamente e la paletta viene ri-riempita.

Dichiarazione

```
class CColorGenerator
```

Titolo

```
#include <Graphics\ColorGenerator.mqh>
```

I metodi della Classe

Metodo	Descrizione
Next	Restituisce il colore successivo dalla tavolozza
Reset	Resetta il generatore

Next

Restituisce il prossimo colore dalla tavolozza.

```
uint  Next()
```

Valore di ritorno

Color.

Nota

Se tutti i colori dalla tavolozza vi sono già passati attraverso, nuovi colori vengono generati automaticamente per sostituire quelli vecchi nella tavolozza.

Reset

Resetta il generatore.

```
void  Reset()
```

CCurve

La classe CCurve funziona con le proprietà delle curve generate sul grafico.

Descrizione

La classe CCurve imposta, installa e riceve le coordinate e le diverse proprietà delle curve quando si lavora con la classe CGraphic.

Ci sono tre modi di tracciamento curve: punti, righe e istogramma. Separati parametri vengono implementati per ogni modalità di compilazione nella classe.

Dichiarazione

```
class CCurve : public CObject
```

Title

```
#include <Graphics\Curve.mqh>
```

Gerarchia ereditaria

CObject

CCurve

Metodi della Classe

Metodo	Descrizione
Tipo	Ottiene il tipo di curva
Name	Ottieni il nome della curva
Color	Ottieni il colore della curva
XMax	Ottiene il valore massimo della funzione X
XMin	Ottiene il valore minimo della funzione X
YMax	Ottiene il valore massimo della funzione Y
YMin	Ottiene il valore minimo della funzione Y
Size	Ottiene il numero di punti che definiscono una curva
PointsSize	Ottiene/imposta la grandezza lineare dei punti che definiscono una curva
PointsFill	Ottiene/imposta la flag per il riempimento di punti che definiscono una curva
PointsColor	Ottiene/imposta il colore di riempimento del punto
GetX	Ottiene i valori X di tutti i punti della curva nell'array
GetY	Ottiene i valori Y di tutti i punti della curva nell'array

Metodo	Descrizione
LinesStyle	Ottiene/imposta uno stile di linea durante la tracciatura di una curva utilizzando le righe
LinesIsSmooth	Ottiene/imposta la flag di smussatura quando viene eseguito il disegno utilizzando le righe
LinesSmoothTension	Ottiene/imposta il parametro di smussatura della curva quando si disegna utilizzando le righe
LinesSmoothStep	Ottiene/imposta la lunghezza delle linee approssimative per la smussatura quando si tracciano le linee
LinesWidth	Ottiene/imposta una larghezza di una linea quando si tracciano curve utilizzando linee
HistogramWidth	Ottiene/imposta la larghezza delle colonne durante la tracciatura utilizzando un istogramma
CustomPlotCBData	Ottiene/imposta il puntatore sull'oggetto da utilizzare nella modalità di disegno curva personalizzata.
CustomPlotFunction	Ottiene/imposta il puntatore alla funzione che implementa la modalità di grafica di curva personalizzata.
PointsType	Ottiene/imposta la flag che punta al tipo di punti utilizzati durante la tracciatura di una curva punteggiata.
StepsDimension	Ottiene/imposta il valore che indica la dimensione utilizzata nel rendering della curva di tipo-step.
TrendLineCoefficients	Ottiene/imposta i rapporti della linea di trend per la scrittura in un array.
TrendLineColor	Ottiene/imposta un colore di una linea di tendenza per una curva.
TrendLineVisible	Ottiene/imposta la flag di visibilità della linea di tendenza.
Update	Aggiorna le coordinate della curva.
Visible	Ottiene/imposta la flag che definisce se una funzione è visibile sul grafico.

Type

Restituire il tipo di curva.

```
ENUM_CURVE_TYPE   Type()
```

Valore di ritorno

Tipo di curva.

Name

Restituisce il nome della curva.

```
string  Name()
```

Valore di ritorno

Nome della curva.

Color

Restituisce il colore della curva.

```
uint  Color()
```

Valore di ritorno

Colore della curva.

XMax

Restituisce il valore massimo della funzione X (solo numeri reali).

```
double  XMax()
```

Valore di ritorno

Massimo numero reale tra tutti gli argomenti della funzione.

XMin

Restituisce il valore massimo della funzione Y (solo numeri reali).

```
double  XMin()
```

Valore di ritorno

Minimo numero reale tra tutti gli argomenti della funzione.

YMax

Restituisce il valore massimo della funzione Y (solo numeri reali).

```
double  YMax()
```

Valore di ritorno

Valore massimo della funzione Y (solo numeri reali).

YMin

Restituisce il valore massimo della funzione Y (solo numeri reali).

```
double  YMin()
```

Valore di ritorno

Valore minimo della funzione Y (solo numeri reali).

Size

Restituisce il numero di punti che definiscono la curva.

```
int  Size()
```

Valore di ritorno

Numero di punti che definiscono la curva.

PointsSize (Metodo Get)

Restituisce la dimensione lineare (in pixel) di punti usati nel disegno della curva.

```
int  PointsSize()
```

Valore di ritorno

Grandezza dei punti che definiscono la curva in pixel.

PointsSize (Metodo Set)

Imposta la dimensione lineare (in pixel) di punti usati nel disegno della curva.

```
void  PointsSize(
   const int  size      // grandezza dei punti in pixels
   )
```

Parametri

size

[in] Grandezza lineare (in pixel) dei punti usati nel disegno della curva.

PointsFill (Metodo Get)

Restituisce un flag che indica se deve essere eseguito un riempimento per i punti che definiscono una curva.

```
bool  PointsFill ()
```

Valore di ritorno

Il valore del flag.

Nota

true — fa il riempimento

false — non fa il riempimento

PointsFill (Metodo Set)

Imposta un flag che indica se deve essere eseguito un riempimento per i punti che definiscono una curva.

```
void  PointsFill(
   const bool  fill      // valore flag
   )
```

Parametri

fill

[in] Valore Flag.

Nota

true — fa il riempimento

false — non fa il riempimento

PointsColor (Metodo Get)

Restituisce il colore dei punti di riempimento.

```
uint  PointsColor ()
```

Valore di ritorno

Colore dei punti di riempimento che definiscono la curva.

PointsColor (Metodo Set)

Imposta il colore di riempimento dei punti

```
void  PointsColor(
   const uint  clr      //colore di riempimento dei punti
   )
```

Parametri

clr

[in]Colore dei punti che definiscono la curva.

GetX

Scrive le coordinate X per tutti i punti curva nell'array.

```
void  GetX(
   double  &x[]      // Coordinate X
   )
```

Parametri

&x[]

[out] Array per la scrittura coordinate X.

GetY

Scrive le coordinate Y per tutti i punti della curva nell'array

```
void  GetY(
   double  &y[]      // Coordinate Y
   )
```

Parametri

&y[]

[out] Array per la scrittura coordinate Y.

LinesStyle (Metodo Get)

Restituisce lo stile della linea quando si disegna una curva utilizzando le linee.

```
ENUM_LINE_STYLE  LinesStyle()
```

Valore di ritorno

Stile della linea.

LinesStyle (Metodo Set)

Imposta lo stile della linea quando si disegna una curva utilizzando le linee.

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // stile della linea
   )
```

Parametri

style

[in] Stile della linea.

LinesIsSmooth (Metodo Get)

Restituisce una flag che definisce se la smussatura dovrebbe essere fatta, quando si disegna una curva per linee.

```
bool  LinesIsSmooth()
```

Valore di ritorno

Valore Flag

Nota

true — fa la smussatura

false — non fa la smussatura

LinesIsSmooth (Metodo Set)

Imposta una flag che definisce se la smussatura dovrebbe essere fatta, quando si disegna una curva per linee.

```
void  LinesIsSmooth(
   const bool  smooth      // valore flag
   )
```

Parametri

smooth

[in] Valore Flag

Nota

true — fa la smussatura

false — non fa la smussatura

LinesSmoothTension (Metodo Get)

Restituisce il parametro di smussatura della curva quando si disegna utilizzando le linee

```
double  LinesSmoothTension()
```

Valore di ritorno

Valore del parametro di smussatura

Nota

Il valore 'tensione' è all'interno del range [0.0 ; 1.0].

LinesSmoothTension (Metodo Set)

Imposta il parametro di smussatura della curva quando si disegna utilizzando le linee

```
void  LinesSmoothTension(
   const double  tension      // vakore del parametro
   )
```

Parametri

tension

[in] Valore del parametro di smussatura.

Nota

Il valore 'tensione' è all'interno del range [0.0 ; 1.0].

LinesSmoothStep (Metodo Get)

Restituisce la lunghezza delle linee approssimanti per la smussatura durante il disegno per linee.

```
double  LinesSmoothStep()
```

Valore di ritorno

Lunghezza di approssimazione linee in pixel.

LinesSmoothStep (Metodo Set)

Imposta la lunghezza delle linee approssimanti per la smussatura durante il disegno per linee.

```
void  LinesSmoothStep(
   const double  step      // lunghezza linea
   )
```

Parametri

step

[in] Lunghezza delle linee approssimanti

LinesEndStyle (Metodo Set)

Ottiene la flag che indica le linee di fine stile di disegno quando utilizza linee per tracciare una curva.

```
ENUM_LINE_END  LinesEndStyle()
```

Return Value

Un valore della flag che indica le linee di fine stile di disegno quando si utilizzano linee per tracciare una curva.

LinesEndStyle (Metodo Get)

Imposta la flag che indica le linee di fine stile di disegno quando si utilizzano linee per tracciare una curva.

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // valore flag
   )
```

Parametri

end_style

[in] Valore della flag che indica l'estremità della linea che traccia lo stile quando si utilizzano linee per tracciare una curva.

LinesWidth (Metodo Get)

Riceve lo spessore delle linee quando traccia una curva utilizzando linee.

```
int  LinesWidth()
```

Return Value

Spessore delle linee.

LinesWidth (Metodo Set)

Imposta lo spessore delle linee quando traccia una curva utilizzando linee.

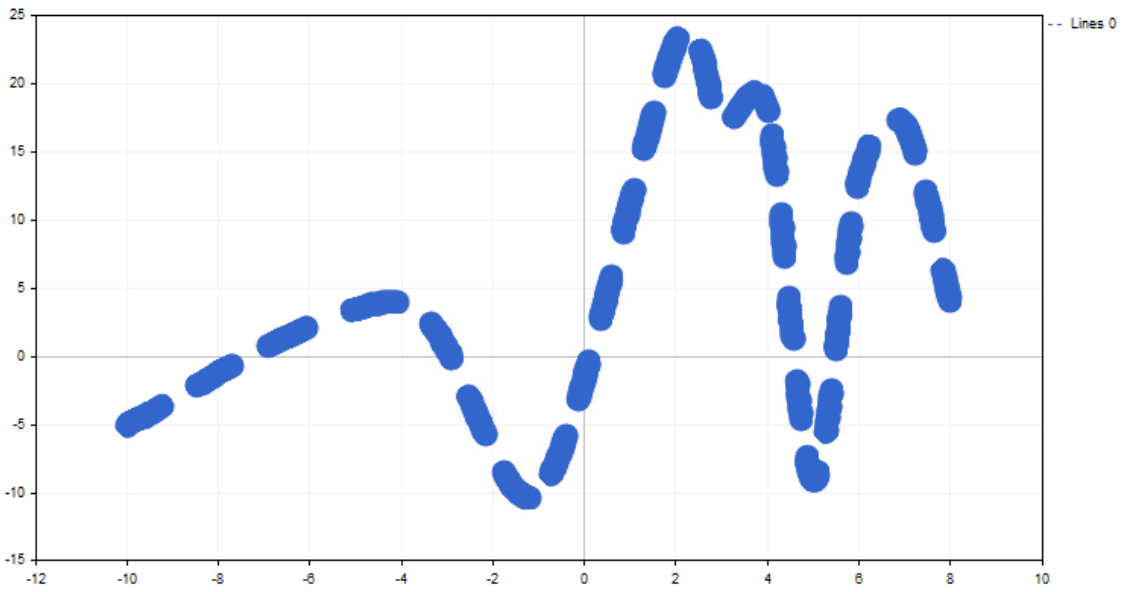
```
void  LinesWidth(
   const int  width      // spessore linee
   )
```

Parametri

width

[in] Spessore linee quando si disegna una curva utilizzando le linee.

Esempio:



La larghezza di una linea è stata modificata utilizzando il seguente codice:

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Funzione start programma Script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]= { -100,-40,-10,20,30,40,50,60,70,80,120 };
   double y[]= { -5,4,-10,23,17,18,-9,13,17,4,9 };
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"ThickLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"ThickLineGraphic");
     }
//--- crea curva
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_LINES);
//--- imposta le proprietà della curva
   curve.LinesSmooth(true);
   curve.LinesStyle(STYLE_DASH);
   curve.LinesEndStyle(LINE_END_ROUND);
   curve.LinesWidth(10);
//--- disegna
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

HistogramWidth (Metodo Get)

Restituisce la larghezza delle colonne durante il disegno utilizzando un istogramma.

```
int  HistogramWidth()
```

Valore di ritorno

La larghezza della colonna in pixel.

HistogramWidth (Metodo Set)

Imposta la larghezza delle colonne durante il disegno utilizzando un istogramma.

```
void  HistogramWidth(
   const int  width      // larghezza colonna
   )
```

Parametri

width

[in] Larghezza della colonna in pixel.

CustomPlotCBData (Metodo Get)

Ottiene il puntatore all'oggetto da utilizzare nella modalità di disegno curva personalizzata.

```
void*  CustomPlotCBData()
```

Return Value

Puntatore all'oggetto per la modellazione curva personalizzata.

CustomPlotCBData (Metodo Set)

Imposta il puntatore sull'oggetto da utilizzare nella modalità di disegno curva personalizzata.

```
void  CustomPlotCBData(
   void*  cbdata      // puntatore all'oggetto
   )
```

Parametri

cbdata

[in] Il puntatore dell'oggetto da utilizzare nella modalità di disegno curva personalizzata

CustomPlotFunction (Metodo Get)

Ottiene il puntatore alla funzione che implementa la curva personalizzata.

```
PlotFucntion  CustomPlotFunction()
```

Return Value

Indicatore della funzione che implementa la modalità di disegno curva personalizzata.

CustomPlotFunction (Metodo Set)

Imposta il puntatore sulla funzione che implementa la modalità di disegno curva personalizzata.

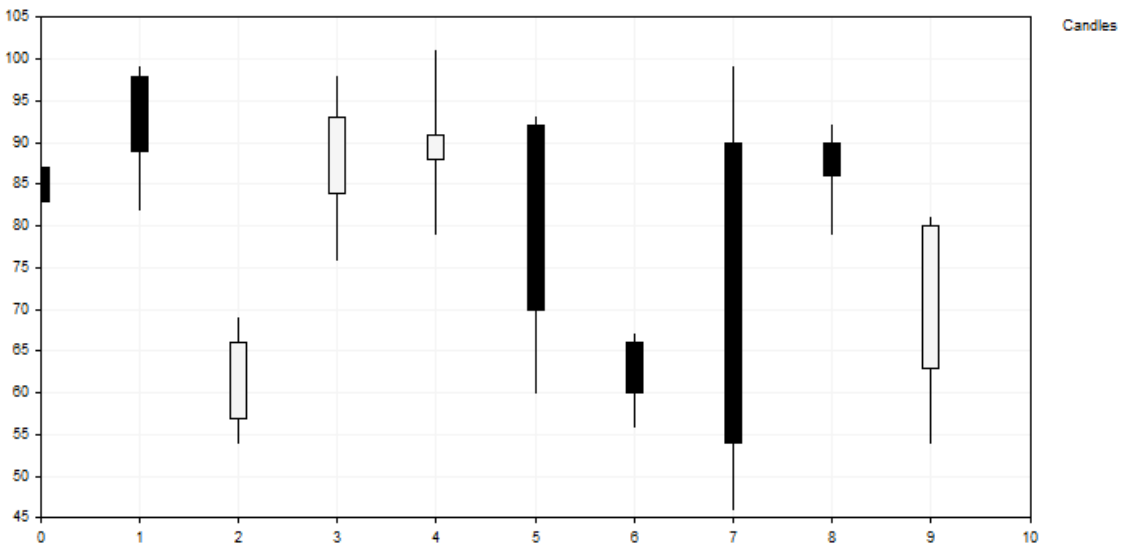
```
void  CustomPlotFunction(
   PlotFucntion  func      // puntatore alla funzione
   )
```

Parametri

func

[in] Puntatore alla funzione che implementa la curva personalizzata

Esempio:



Questa curva costituita da barre è costruita utilizzando il seguente codice:

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Classe CCandle                                                   |
//| Uso: classe per rappresentare la candela                         |
//+------------------------------------------------------------------+
class CCandle: public CObject
  {
private:
   double            m_open;
   double            m_close;
   double            m_high;
   double            m_low;
   uint              m_clr_inc;
   uint              m_clr_dec;
   int               m_width;

public:
                     CCandle(const double open,const double close,const double high,co
                                                    const int width,const uint clr_
                    ~CCandle(void);
   double            OpenValue(void)             const { return(m_open);     }
   double            CloseValue(void)            const { return(m_close);    }
   double            HigthValue(void)            const { return(m_high);     }
   double            LowValue(void)              const { return(m_low);      }
   uint              CandleColorIncrement(void)  const { return(m_clr_inc);  }
   uint              CandleColorDecrement(void)  const { return(m_clr_dec);  }
   int               CandleWidth(void)           const { return(m_width);    }
  };
//+------------------------------------------------------------------+
//| Costruttore                                                      |
//+------------------------------------------------------------------+
CCandle::CCandle(const double open,const double close,const double high,const double l
                                 const int width,const uint clr_inc=0x000000,const uin
                                 m_open(open),m_close(close),m_high(high),m_low(low),
                                 m_clr_inc(clr_inc),m_clr_dec(clr_dec),m_width(width)
  {
  }
//+------------------------------------------------------------------+
//| Distruttore                                                      |
//+------------------------------------------------------------------+
CCandle::~CCandle(void)
  {
  }
//+------------------------------------------------------------------+
//| Metodo Custom per disegnare candele                              |
//+------------------------------------------------------------------+
void PlotCandles(double &x[],double &y[],int size,CGraphic *graphic,CCanvas *canvas,vo
  {
//--- check obj
   CArrayObj *candles=dynamic_cast<CArrayObj*>(cbdata);
   if(candles==NULL || candles.Total()!=size)
      return;
//--- plot candles
   for(int i=0; i<size; i++)
     {
      CCandle *candle=dynamic_cast<CCandle*>(candles.At(i));
```

```
      if(candle==NULL)
         return;
      //--- calcolo primario
      int xc=graphic.ScaleX(x[i]);
      int width_2=candle.CandleWidth()/2;
      int open=graphic.ScaleY(candle.OpenValue());
      int close=graphic.ScaleY(candle.CloseValue());
      int high=graphic.ScaleY(candle.HigthValue());
      int low=graphic.ScaleY(candle.LowValue());
      uint clr=(open<=close) ? candle.CandleColorIncrement() :  candle.CandleColorDeci
      //--- disegna candela
      canvas.LineVertical(xc,high,low,0x000000);
      //--- disegna il corpo real della
      canvas.FillRectangle(xc+width_2,open,xc-width_2,close,clr);
      canvas.Rectangle(xc+width_2,open,xc-width_2,close,0x000000);
     }
  }
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int count=10;
   int width=10;
   double x[];
   double y[];
   ArrayResize(x,count);
   ArrayResize(y,count);
   CArrayObj candles();
   double max=0;
   double min=0;
//--- crea valori
   for(int i=0; i<count; i++)
     {
      x[i] = i;
      y[i] = i;
      //--- calcola valori
      double open=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double close=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double high=MathRound(MathMax(open,close)+(MathRand()/32767.0)*10.0);
      double low=MathRound(MathMin(open,close) -(MathRand()/32767.0)*10.0);
      //--- trova max e min
      if(i==0 || max<high)
         max=high;
      if(i==0 || min>low)
         min=low;
      //--- crea candela
      CCandle *candle=new CCandle(open,close,high,low,width);
      candles.Add(candle);
     }
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"CandleGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"CandleGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_CUSTOM,"Candles");
//--- imposta le proprietà della
   curve.CustomPlotFunction(PlotCandles);
   curve.CustomPlotCBData(GetPointer(candles));
```

```
//--- imposta le proprietà grafiche
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- disegna
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

PointsType (Metodo Get)

Ottiene la flag che punta al tipo di punti utilizzati durante la tracciatura di una curva punteggiata.

```
ENUM_POINT_TYPE  PointsType()
```

Return Value

Il valore della flag che indica il tipo di punti.

PointsType (Metodo Set)

Imposta la flag che punta al tipo di punti utilizzati durante la tracciatura di una curva punteggiata.

```
void  PointsType(
   ENUM_POINT_TYPE  type      // valori flag
   )
```

Parametri

type

[in] Il valore della flag che punta al tipo di punti utilizzati durante la tracciatura di una curva punteggiata.

StepsDimension (Metodo Get)

Ottiene il valore che indica la dimensione utilizzata nel rendering della curva a tipo-step.

```
int  StepsDimension()
```

Return Value

Dimensione utilizzata nel rendering della curva a step.

StepsDimension (Metodo Set)

Imposta il valore che indica la dimensione utilizzata per il rendering di una curva a tipo-step.

```
void  StepsDimension(
   const int  dimension      // dimensione
   )
```

Parametri

dimension

[in] Dimensione (0 o 1).

Nota

0 — x (la linea orizzontale è seguita da quella verticale).

1 — y (la linea verticale è seguita da quella orizzontale).

TrendLineCoefficients (Metodo Get)

Ottiene rapporti della linea di trend per la scrittura in un array.

```
double&  TrendLineCoefficients()
```

Return Value

Rapporti della linea di tendenza.

TrendLineCoefficients (Metodo Set)

Imposta i rapporti della linea di tendenza per la scrittura in un array.

```
void  TrendLineCoefficients(
   double&  coefficients[]      // array per scrivere rapporti
   )
```

Parametri

coefficients[]

[out] Array per scrivere rapporti.

TrendLineColor (Metodo Get)

Ottiene il colore di una linea di tendenza per una curva.

```
uint  TrendLineColor()
```

Return Value

Colore della linea di tendenza.

TrendLineColor (Metodo Set)

Imposta il colore di una linea di tendenza per una curva.

```
void  TrendLineColor(
   const uint  clr      // colore della trend line
   )
```

Parametri

clr

[in] Colore Linea

TrendLineVisible (Metodo Get)

Ottiene la flag di visibilità della linea di tendenza.

```
bool  TrendLineVisible()
```

Return Value

Il valore della flag che specifica se è visibile una linea di tendenza.

TrendLineVisible (Metodo Set)

Imposta la barra di visibilità della linea di tendenza.

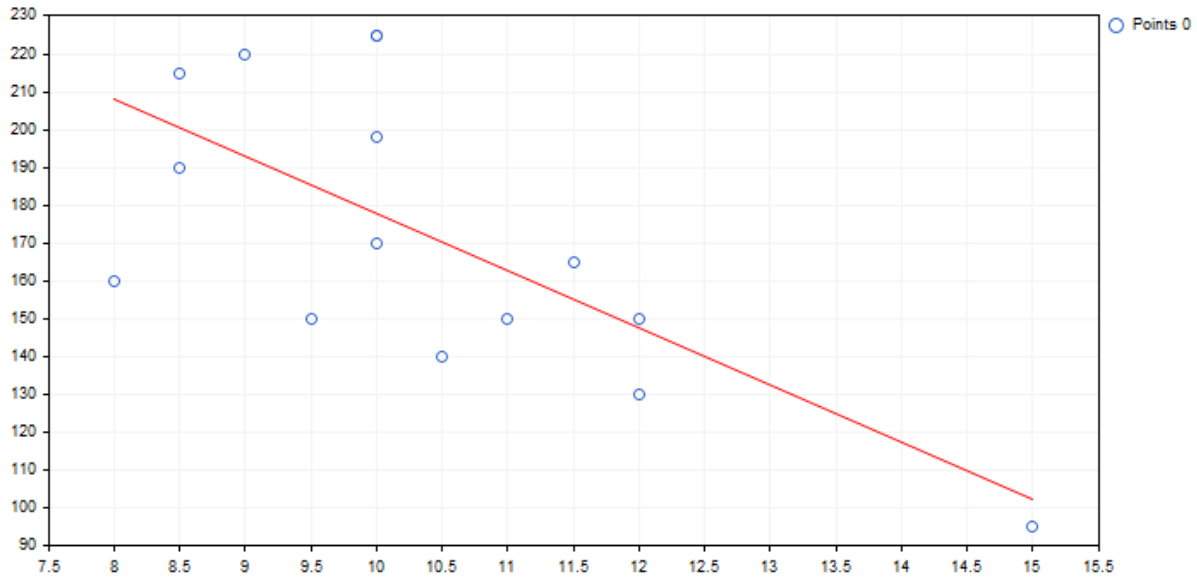
```
void  TrendLineVisible(
   const bool  visible      // valore flag
   )
```

Parametri

visible

[in] Il valore della flag di visibilità della linea di tendenza.

Esempio:



Di seguito è riportato il codice della linea di tendenza menzionata e il suo plotting sul grafico:

```
//+------------------------------------------------------------------+
//|                                              TrendLineGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Funzione start del programma Script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]={12.0,11.5,11.0,12.0,10.5,10.0,9.0,8.5,10.0,8.5,10.0,8.0,9.5,10.0,15.0};
   double y[]={130.0,165.0,150.0,150.0,140.0,198.0,220.0,215.0,225.0,190.0,170.0,160.0
//--- crea grafica
   CGraphic graphic;
   if(!graphic.Create(0,"TrendLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"TrendLineGraphic");
     }
//--- crea curva
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_POINTS);
//--- imposta le proprietà della curva
   curve.TrendLineVisible(true);
   curve.TrendLineColor(ColorToARGB(clrRed));
//--- disegna
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

Update

Aggiorna le coordinate della curva.

La versione per il lavoro da coordinate Y. Gli indici dell'array passati vengono utilizzati come coordinate X qui.

```
void  Update(
   const double&  y[]   // coordinate Y
   )
```

Questa versione utilizza le coordinate X ed Y.

```
void  Update(
   const double&  x[],     // coordinate X
   const double&  y[]      // coordinate Y
   )
```

La versione per lavorare con punti CPoint2D.

```
void  Update(
   const CPoint2D&  points[]      // Coordinate della Curva
   )
```

La versione per lavorare con un puntatore alla funzione CurveFunction.

```
void  Update(
   CurveFunction  function,     // puntatore alla funzione descrivente una curva
   const double   from,         // valore iniziale dell'argomento della funzione
   const double   to,           // valore finale dell'argomento della funzione
   const double   step          // incremento argomento
   )
```

Parametri

x[]

[in] coordinate X.

y[]

[in] coordinate Y.

points[]

[in] Coordinate della Curva.

function

[in] Un puntatore alla funzione che descrive una curva

from

[in] Valore iniziale dell'argomento della funzione

to

[in] Valore finale dell'argomento della funzione

step

[in] Incremento argomento

Visible (Metodo Get)

Ottiene la flag che definisce se una funzione è visibile sul grafico.

```
void  Visible(
   const bool  visible      //
   )
```

Return Value

Un valore della flag che definisce la visibilità della funzione sul grafico.

Visible (Metodo Set)

Imposta la flag che definisce se una funzione è visibile sul grafico.

```
void  Visible(
   const bool  visible      // valore flag
   )
```

Parametri

visible

[in] Un valore della flag che definisce la visibilità della funzione sul grafico.

CGraphic

CGraphic è una classe di base per la creazione di grafici personalizzati.

Descrizione

La classe CGraphic offre numerosi aspetti di lavoro con grafici personalizzati.

La classe memorizza gli elementi principali del chart, imposta i loro parametri ed esegue il plotting.

Inoltre, la classe memorizza le curve per il chart e fornisce varie opzioni di visualizzazione.

Dichiarazione

```
class CGraphic
```

Title

```
#include <Graphics\Graphic.mqh>
```

Metodi della Classe

Metodo	Descrizione
Create	Creare una risorsa grafica legata ad un oggetto chart
Destroy	Rimuovere un chart e distruggere una risorsa grafica
Update	Visualizzare le modifiche apportate
ChartObjectName	Ottiene il nome di un oggetto associato ad un chart
ResourceName	Ottiene il nome della risorsa grafica
XAxis	Ottiene il puntatore sull'asse X
YAxis	Prende il puntatore sull'asse Y
GapSize	Ottiene/imposta la grandezza degli indentatori tra gli elementi del chart
BackgroundColor	Ottiene/imposta un colore di sfondo
BackgroundMain	Ottiene/imposta un'intestazione di tabella
BackgroundMainSize	Ottiene/imposta la dimensione del carattere del sotto-header
BackgroundMainColor	Ottiene/imposta un colore dell'intestazione del chart
BackgroundSub	Ottiene/imposta una sotto-intestazione
BackgroundSubSize	Ottiene/imposta la dimensione del carattere del sotto-header
BackgroundSubColor	Ottiene/imposta un colore della sotto-intestazione del chart

Metodo	Descrizione
GridLineColor	Ottiene/imposta un colore della linea della griglia
GridBackgroundColor	Ottiene/imposta un colore di sfondo della griglia
GridCircleRadius	Ottiene/imposta il raggio di punti nei nodi della griglia
GridCircleColor	Ottiene/imposta il colore del punto nei nodi della griglia
GridHasCircle	Ottiene/imposta la flag di plotting dei punti nei nodi della griglia
GridAxisLineColor	Ottiene il valore di un colore reale degli assi del chart.
HistoryNameWidth	Ottiene/imposta la lunghezza massima consentita per visualizzare il nome di una curva
HistoryNameSize	Ottiene/imposta la dimensione del carattere del nome di una curva
HistorySymbolSize	Ottiene/imposta la dimensione dei simboli convenzionali notazionali
TextAdd	Aggiunge un testo al chart
LineAdd	Aggiungere una riga al chart
CurveAdd	Creare ed aggiunge una curva al chart
CurvePlot	Traccia una curva precedentemente creata per indice
CurvePlotAll	Traccia tutte le curve precedentemente create
CurveGetByIndex	Ottiene una curva da un indice specificato
CurveGetByName	Ottiene una curva da un nome specificato
CurveRemoveByIndex	Rimuovere una curva per indice specificato.
CurveRemoveByName	Rimuove una curva da un nome specificato.
CurvesTotal	Prendi il numero di curve per il chart dato.
MarksToAxisAdd	Aggiunge un segno di scala all'asse del chart
MajorMarkSize	Ottiene/imposta la grandezza della scala dei ticks sull'asse del chart
FontSet	Imposta i parametri del carattere corrente
FontGet	Ottiene i parametri del carattere corrente
Attach	Ottiene/imposta una risorsa grafica e la lega all'istanza della classe CGraphic

Metodo	Descrizione
CalculateMaxMinValues	Calcola(ricalcola) i valori chart minimo e massimo su entrambi gli assi.
Altezza	Ottiene l'altezza del chart in pixel.
IndentDown	Ottiene/imposta un'indentazione del chart dal bordo inferiore.
IndentLeft	Ottiene/imposta un'indentazone del chart dal bordo sinistro.
IndentRight	Ottiene/imposta un'indentazone del chart dal bordo destro.
IndentUp	Ottiene/imposta un'indentazone del chart dal bordo superiore.
Redraw	Ridisegna il chart.
ResetParameters	Reimposta i parametri di ridisegnamento del chart.
ScaleX	Scala il valore per asse X.
ScaleY	Scala il valore con l'asse Y.
SetDefaultParameters	Imposta i parametri del grafico sui valori predefiniti.
Width	Ottiene la larghezza del grafico in pixel.

Create

Crea una risorsa grafica legata ad un oggetto chart.

```
bool  Create(
   const long     chart,      // chart ID
   const string   name,       // nome
   const int      subwin,     // indice sotto-finestra
   const int      x1,         // x1 coordinate
   const int      y1,         // y1 coordinate
   const int      x2,         // x2 coordinate
   const int      y2          // y1 coordinate
   )
```

Parametri

chart

[in] Chart ID.

name

[in] Nome.

subwin

[in] Indice Sub-window.

x1

[in] X1 coordinate.

y1

[in] Y1 coordinate.

x2

[in] X2 coordinate.

y2

[in] Y2 coordinate.

Destroy

Rimuove un chart e distrugge una risorsa grafica.

```
void  Destroy()
```

Aggiorna

Visualizza le modifiche implementate.

```
void  Update(
   const bool  redraw=true      // flag
   )
```

Parametri

redraw=true

[in] Valore Flag

ChartObjectName

Ottiene il nome di un oggetto legato al chart.

```
string  ChartObjectName()
```

Valore di ritorno

Nome di un oggetto legato al chart.

ResourceName

Riceve il nome di una risorsa grafica.

```
string  ResourceName()
```

Valore di ritorno

Nome di una risorsa grafica.

XAxis

Restituisce il puntatore per l'asse X.

```
CAxis  *XAxis()
```

Valore di ritorno

Puntatore all'asse X.

YAxis

Restituisce il puntatore per l'asse y.

```
CAxis  *YAxis()
```

Valore di ritorno

Puntatore all'asse Y.

GapSize (Metodo Get)

Restituisce la dimensione dei rientri tra gli elementi del chart.

```
int  GapSize()
```

Valore di ritorno

Dimensione rientro in pixel.

GapSize (Metodo Set)

Imposta la dimensione dei rientri tra gli elementi del chart.

```
void  GapSize(
   const int  size     // grandezza del rientro
   )
```

Parametri

size

[in] Dimensioni rientro in pixel.

BackgroundColor (Metodo Get)

Restituisce il colore di sfondo.

```
color  BackgroundColor()
```

BackgroundColor (Metodo Set)

Imposta il colore di sfondo.

```
void  BackgroundColor(
   const color  clr      // colore di sfondo
   )
```

Parametri

clr

[in] Colore di sfondo.

BackgroundMain (Metodo Get)

Restituisce un header del chart

```
string  BackgroundMain()
```

BackgroundMain (Metodo Set)

Imposta un testo di intestazione del chart.

```
void  BackgroundMain(
   const string  main      // testo dell'header(intestazione)
   )
```

Parametri

main

[in] Testo dell header del chart

BackgroundMainSize (Metodo Get)

Restituisce la grandezza dell' intestazione.

```
int  BackgroundMainSize()
```

Valore di ritorno

Dimensione del carattere dell header.

BackgroundMainSize (Metodo Set)

Imposta la dimensione dell' intestazione.

```
void  BackgroundMainSize(
   const int  size     // grandezza header
   )
```

Parametri

size

[in] Grandezza del font dell header.

BackgroundMainColor (Metodo Get)

Restituisce il colore dell' intestazione.

```
color  BackgroundMainColor()
```

Valore di ritorno

Colore dell'intestazione(header).

BackgroundMainColor (Metodo Set)

Imposta il colore dell header.

```
void  BackgroundMainColor(
   const color  clr      // colore dell header
   )
```

Parametri

clr

[in] Colore dell Header.

BackgroundSub (Metodo Get)

Restituisce il sub-header.

```
string  BackgroundSub()
```

Valore di ritorno

Testo del Sub-header.

BackgroundSub (Metodo Set)

Imposta il testo del sub-header.

```
void  BackgroundSub (Set method)(
   const string  sub       // testo sub-header
   )
```

Parametri

sub

[in] Testo Sub-header.

BackgroundSubSize (Metodo Get)

Restituisce la dimensione del carattere del sub-header

```
int BackgroundSubSize()
```

BackgroundSubSize (Metodo Get)

Imposta la dimensione del sub-header

```
void  BackgroundSubSize(
   const int  size      // grandezza del carattere del sub-header
   )
```

Parametri

size

[in] Grandezza del carattere del Sub-header

BackgroundSubColor (Metodo Get)

Restituisce il colore del sub-header.

```
color  BackgroundSubColor()
```

BackgroundSubColor (Metodo Set)

Imposta il colore del sub-header.

```
void  BackgroundSubColor(
   const color  clr      // colore del sub-header
   )
```

Parametri

clr

[in] Colore Sub-header.

GridLineColor (Metodo Get)

Restituisce il colore della linea della griglia

```
color  GridLineColor()
```

Valore di ritorno

colore della linea della griglia.

GridLineColor (Metodo Set)

Imposta il colore della linea della griglia.

```
void  GridLineColor(
   const color  clr      // colore della linea
   )
```

Parametri

clr

[in] Colore della linea della griglia.

GridBackgroundColor (Metodo Get)

Restituisce il colore della griglia di sfondo

```
color  GridBackgroundColor()
```

Valore di ritorno

Colore di sfondo della griglia

GridBackgroundColor (Metodo Set)

Imposta il colore di sfondo della griglia

```
void  GridBackgroundColor(
   const color  clr      // colore di sfondo della griglia
   )
```

Parametri

clr

[in] Colore di sfondo della griglia

GridCircleRadius (Metodo Get)

Restituisce il raggio dei punti nei nodi della griglia.

```
int  GridCircleRadius()
```

Valore di ritorno

Raggio punti in pixel.

GridCircleRadius (Metodo Set)

Imposta il raggio dei punti nei nodi della griglia

```
void  GridCircleRadius(
   const int  r      // raggio
   )
```

Parametri

r

[in] Raggio dei punti in pixels.

GridCircleColor (Metodo Get)

Restituisce il colore dei punti nei nodi della griglia.

```
color  GridCircleColor()
```

Valore di ritorno

Colore dei punti.

GridCircleColor (Metodo Set)

Imposta il colore di punti nei nodi della griglia.

```
void  GridCircleColor(
   const color  clr      // colore dei punti
   )
```

Parametri

clr

[in] Colore dei punti.

GridHasCircle (Metodo Get)

Restituisce la flag che definisce se i punti nei nodi della griglia devono essere visualizzati.

```
bool  GridHasCircle()
```

Valore di ritorno

Il valore del flag.

Nota

true — mostra i punti

false — non mostra i punti

GridHasCircle (Metodo Set)

Imposta la flag che definisce se i punti nei nodi della griglia devono essere visualizzati.

```
void  GridHasCircle(
   const bool  has
   )
```

Parametri

has

[in] Valore Flag.

Nota

true — mostra i punti

false — non mostra i punti

GridAxisLineColor (metodo Get)

Ottiene il valore di un colore reale degli assi del chart.

```
uint  GridAxisLineColor()
```

Return Value

Colore degli assi del chart.

GridAxisLineColor (metodo Set)

Imposta il valore di colore asse reale del chart.

```
void  GridAxisLineColor(
   const uint  clr      // colori assi del chart
   )
```

Parametri

clr

[in] Colore degli assi reali del chart.

HistoryNameWidth (Metodo Get)

Restituisce la lunghezza massima consentita per la visualizzazione del nome della curva.

```
int  HistoryNameWidth()
```

Valore di ritorno

Lunghezza massima in pixel.

Nota

Se il nome della curva supera la lunghezza massima consentita, è troncato e punti vengono aggiunti alla sua fine.

HistoryNameWidth (Metodo Set)

Imposta la lunghezza massima consentita per la visualizzazione del nome della curva.

```
void  HistoryNameWidth(
   const int  width      // lunghezza massima
   )
```

Parametri

width

[in] Lunghezza massima in pixels.

Nota

Se il nome della curva supera la lunghezza massima consentita, è troncato e punti vengono aggiunti alla sua fine.

HistoryNameSize (Metodo Get)

Restituisce la dimensione del carattere del nome della curva.

```
int  HistoryNameSize()
```

Valore di ritorno

Dimensione del carattere del nome della curva.

HistoryNameSize (Metodo Set)

Imposta la dimensione del carattere del nome della curva.

```
void  HistoryNameSize (Set method)(
   const int  size      // gradezza del nome del font
   )
```

Parametri

size

[in] Grandezza del nome del carattere.

HistorySymbolSize (Metodo Get)

Restituisce la dimensione dei simboli convenzione di notazione del chart

```
int  HistorySymbolSize()
```

Valore di ritorno

Dimensioni dei simboli di convenzione di notazione

HistorySymbolSize (Metodo Set)

Imposta la dimensione dei simboli convenzione di notazione del chart

```
void  HistorySymbolSize(
   const int  size      // grandezza simbolo
   )
```

Parametri

size

[in] Dimensioni dei simboli di convenzione di notazione.

TextAdd

Aggiunge un testo al chart.

Versione per lavorare con coordinate X e Y

```
void  TextAdd(
   const int     x,              // X coordinate
   const int     y,              // Y coordinate
   const string  text,           // testo
   const uint    clr,            // colore
   const uint    alignment=0     // allineamento
   )
```

Versione per CPoint

```
void  TextAdd(
   const CPoint  &point,         // coordinate del punto
   const string  text,           // testo
   const uint    clr,            // colore
   const uint    alignment=0     // allineamento
   )
```

Parametri

x

[in] X coordinate.

y

[in] Y coordinate.

&point

[in] Coordinate del punto.

text

[in] Text.

clr

[in] Colore.

alignment=0

[in] Allineamento.

LineAdd

Aggiunge una linea al chart.

Questa versione utilizza coordinate X e Y

```
void  LineAdd(
   const int   x1,         // x1 coordinate
   const int   y1,         // y1 coordinate
   const int   x2,         // x2 coordinate
   const int   y2,         // y2 coordinate
   const uint  clr,        // colore
   const uint  style       // stile
   )
```

Versione per CPoint

```
void  LineAdd2(
   const CPoint  &point1,     // coordinate primo punto
   const CPoint  &point2,     // coordinate secondo punto
   const uint    clr,         // colore
   const uint    style        // stile
   )
```

Parametri

x1

[in] X1 coordinate.

y1

[in] Y1 coordinate.

x2

[in] X2 coordinate.

y2

[in] Y2 coordinate.

&point1

[in] Coordinate primo punto.

&point2

[in] coordinate secondo punto.

clr

[in] Colore.

style

[in] Stile.

CurveAdd

Creare ed aggiunge una nuova curva al chart.

Questa versione utilizza la coordinata Y (un colore curva è impostato automaticamente)

```
CCurve* CurveAdd(
   const double      &y[],          // Coordinate Y
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

Nota

Indici di array Y sono utilizzati come coordinate X per la curva.

Questa versione utilizza le coordinate X ed Y (un colore curva è impostato automaticamente)

```
CCurve*  CurveAdd(
   const double      &x[],          // Coordinate X
   const double      &y[],          // Coordinate Y
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

La versione per lavorare con i punti CPoint2D (il colore della curva viene impostato automaticamente)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordinate punto
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

La versione per lavorare con il puntatore alla funzione CurveFunction (il colore della curva viene impostato automaticamente)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // puntatore alla funzione
   const double      from,          // valore iniziale dell'argomento
   const double      to,            // valore finale dell'argomento
   const double      step,          // incremento per argomento
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

Versione per lavorare con la coordinata Y (un colore curva è impostato dall' utente)

```
CCurve*  CurveAdd(
   const double      &y[],          // Coordinate Y
   const uint        clr,           // colore curva
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

Nota

Indici di array Y sono utilizzati come coordinate X per la curva.

Questa versione utilizza le coordinate X ed Y (il colore curva è impostato dall'utente)

```
CCurve*  CurveAdd(
   const double      &x[],          // Coordinate X
   const double      &y[],          // Coordinate Y
   const uint        clr,           // colore curva
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

La versione per lavorare con puntini CPoint2D (il colore della curva è impostato dall'utente)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordinate punto
   const uint        clr,           // colore curva
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

Versione per lavorare con il puntatore alla funzione CurveFunction (il colore della curva è impostato dall'utente)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // puntatore alla funzione
   const double      from,          // valore iniziale dell'argomento
   const double      to,            // valore finale dell'argomento
   const double      step,          // incremento per argomento
   const uint        clr,           // colore curva
   ENUM_CURVE_TYPE   type,          // tipo di curva
   const string      name=NULL      // nome della curva
   )
```

Parametri

&x[]

[in] Coordinate X.

&y[]

[in] Coordinate Y.

&points[]

[in] Coordinate per punti.

function

[in] Puntatore alla funzione.

from

[in] Valore iniziale dell'argomento.

to

[in] Valore finale dell'argomento.

step

[in] Incremento per argomento.

type

[in] Tipo di curva.

name=NULL

[in] Nome curva.

clr

[in] Colore curva.

Valore di ritorno

Puntatore alla curva creata.

CurvePlot

Visualizza la curva creata in precedenza con un indice specificato.

```
bool  CurvePlot(
   const int  index      // indice(index)
   )
```

Parametri

index

[in] Indice della Curva

Valore di ritorno

true - successo, altrimenti - false.

CurvePlotAll

Visualizza tutte le curve precedentemente aggiunte al chart.

```
bool  CurvePlotAll()
```

Valore di ritorno

true - successo, altrimenti - false.

CurveGetByIndex

Ottiene la curva da un indice specificato.

```
CCurve*  CurveGetByIndex(
   const int  index      // indice della curva
   )
```

Parametri

index

[in] Indice della curva.

Valore di ritorno

Puntatore alla curva con un indice specificato.

CurveGetByName

Ottiene una curva con un nome specificato.

```
CCurve*  CurveGetByName(
   const string  name      // nome della curva
   )
```

Parametri

name

[in] Nome della Curva.

Valore di ritorno

Puntatore alla prima curva trovata con un nome specificato.

CurveRemoveByIndex

Rimuovere una curva per indice specificato.

```
bool  CurveRemoveByIndex(
   const int  index      // indice curva
   )
```

Parametri

index

[in] Indice della curva da rimuovere.

Return Value

true — successo, altrimenti — false.

CurveRemoveByName

Rimuove una curva da un nome specificato.

```
bool  CurveRemoveByName(
   const string  name      // nome curva
   )
```

Parametri

name

[in] Nome della curva da rimuovere.

Return Value

true — successo, altrimenti — false.

CurvesTotal

Prendi il numero di curve per il chart dato.

```
int  CurvesTotal()
```

Return Value

Numero di curve.

Nota

Tutte le curve sul chart corrente sono considerate indipendentemente dal disegno e visibilità stile.

MarksToAxisAdd

Aggiunge un segno di scala (ticks) all'asse del chart specificato.

```
bool  MarksToAxisAdd(
   const double         &marks[],        // coordinate tick
   const int            mark_size,       // grandezza tick
   ENUM_MARK_POSITION   position,        // posizione tick
   const int            dimension=0      // dimensione
   )
```

Parametri

&marks[]

[in] Coordinate tick

mark_size

[in] Grandezza tick

position

[in] Posizione tick

dimension=0

[in] 0 — aggiunge all'asse X,

1 — aggiunge all'asse Y

Valore di ritorno

true - successo, altrimenti - false.

MajorMarkSize (Metodo Get)

Restituisce la dimensione della scala dei ticks sugli assi delle coordinate.

```
int  MajorMarkSize()
```

MajorMarkSize (Metodo Set)

Restituisce la dimensione della scala dei ticks sugli assi delle coordinate.

```
void  MajorMarkSize(
   const int  size      // grandezza tick
   )
```

Parametri

size

[in] Grandezza tick in pixels.

FontSet

Imposta i parametri del carattere corrente.

```
bool  FontSet(
   const string  name,         // nome
   const int     size,         // grandezza
   const uint    flags=0,      // flags
   const uint    angle=0       // angolo
   )
```

Parametri

name

[in] Nome.

size

[in] Grandezza.

flags=0

[in] Flags.

angle=0

[in] Angolo.

Valore di ritorno

true - successo, altrimenti - false.

FontGet

Ottiene i parametri del carattere corrente.

```
void  FontGet(
   string  &name,       // nome
   int     &size,       // grandezza
   uint    &flags,      // flags
   uint    &angle       // angolo
   )
```

Parametri

&name

[out] Nome.

&size

[out] Grandezza.

&flags

[out] Flags.

&angle

[out] Angolo.

Attach

La versione per ottenere una risorsa grafica dall'oggetto OBJ_BITMAP_LABEL e la lega all'istanza della classe CGraphic:

```
bool  Attach(
   const long     chart_id,      // chart ID
   const string   objname        // nome oggetto della classe
   )
```

La versione per la creazione di una risorsa grafica per l'oggetto OBJ_BITMAP_LABEL e l'associazione all'istanza della classe CGraphic:

```
bool  Attach(
   const long     chart_id,      // chart ID
   const string   objname,       // nome oggetto della classe
   const int      width,         // spessore immagine
   const int      height         // altezza immagine
   )
```

Parametri

chart_id

[in] Chart ID.

objname

[in] Nome dell'ggetto grafico.

width

[in] Spessore immagine nella risorsa.

height

[in] Altezza immagine nella risorsa.

Return Value

true — riuscito, false - non è riuscito a legare l'oggetto.

CalculateMaxMinValues

Calcola(ricalcola) i valori chart minimo e massimo su entrambi gli assi.

```
void  CalculateMaxMinValues()
```

Altezza

Ottiene l'altezza del chart in pixel.

```
int  Height()
```

Return Value

Altezza del chart in pixel.

IndentDown (metodo Get)

Ottiene un rientro del chart dal bordo inferiore.

```
int  IndentDown()
```

Return Value

Grandezza indentazione in pixel.

IndentDown (Metodo Set)

Imposta un rientro(indent) del chart dal bordo inferiore.

```
void  IndentDown(
   const int  down      // grandezza del rientro
   )
```

Parametri

down

[in] Grandezza del rientro in pixels.

IndentLeft (Metodo Get)

Ottiene un rientro del chart dal riquadro di sinistra.

```
int  IndentLeft()
```

Return Value

Grandezza indentazione in pixel.

IndentLeft (Metodo Set)

Imposta un indent del chart dal bordo sinistro.

```
void  IndentLeft(
   const int  left      // grandezza indent
   )
```

Parametri

left

[in] Grandezza del rientro in pixels.

IndentRight (Metodo Get)

Ottiene un indent del chart dal riquadro destro.

```
int  IndentRight()
```

Return Value

Grandezza indentazione in pixel.

IndentRight (Metodo Set)

Imposta un indent del chart dal bordo destro.

```
void  IndentRight(
   const int  right      // grandezza indent
   )
```

Parametri

right

[in] Grandezza del rientro in pixels.

IndentUp (Metodo Get)

Ottiene un'indentazone del chart dal bordo superiore.

```
int  IndentUp()
```

Return Value

Grandezza indentazione in pixel.

IndentUp (Metodo Set)

Imposta un'indentazone del chart dal bordo superiore.

```
void  IndentUp(
   const int  up      // grandezza del rientro
   )
```

Parametri

up

[in] Valore di indentazione in pixel.

Redraw

Ridisegna il chart.

```
bool  Redraw(
   const bool  rescale=false      // valore della flag
   )
```

Parametri

rescale=false

[in] La flag che indica se un grafico dovrebbe essere ridimensionato.

Return Value

true — riuscito, altrimenti — false.

ResetParameters

Reimposta i parametri di ridisegnamento del chart.

```
void  ResetParameters()
```

ScaleX

Scala il valore per asse X.

```
virtual int  ScaleX(
   double  x      // valore per asse X
   )
```

Parametri

x

[in] Valore reale per asse X.

Return Value

Il valore in pixel.

Nota

Scala un valore reale in pixel per la visualizzazione sul chart.

ScaleY

Scala il valore con l'asse Y.

```
virtual int  ScaleY(
   double  y      // valore per asse Y
   )
```

Parametri

y

[in] Valore reale per asse Y.

Return Value

Il valore in pixel.

Nota

Scala un valore reale in pixel per la visualizzazione sul chart.

SetDefaultParameters

Imposta i parametri del grafico sui valori predefiniti.

```
void  SetDefaultParameters()
```

Width

Ottiene la larghezza del grafico in pixel.

```
int  Width()
```

Return Value

Spessore del chart in pixel.

Indicatori tecnici e TimeSeries

Questa sezione contiene i dettagli tecnici delle classi di indicatore tecnico e TimeSeries e la descrizione delle corrispondenti componenti della libreria MQL5 standard.

L'uso delle classi degli indicatori tecnici e TimeSeries farà risparmiare tempo nello sviluppo di applicazioni (script, Expert Advisors).

La MQL5 standard Library (in termini di indicatori tecnici e timeseries) si trova nella directory di lavoro del terminale nella cartella \Include\Indicatori.

Classe/gruppo	Descrizione
Classi Base	Gruppo di classi base ed ausiliari
Classi Timeseries	Gruppo di classi timeseries
Indicatori di Trend	Gruppo di classi Indicatori di tendenza (Trend)
Oscillators	Gruppo di classi indicatore Oscillator
Volume Indicators	Gruppo di classi Indicatori del Volume
Bill Williams Indicators	Gruppo di classi di indicatori Bill Williams
Custom indicators (Indicatori Personalizzati)	Classe indicatore personalizzate

Classi base ed ausiliarie dell indicatore tecnico e timeseries

Questa sezione contiene i dettagli tecnici di classi base ed ausiliarie di indicatore tecnico e TimeSeries e la descrizione delle corrispondenti componenti della libreria standard MQL5.

Classe/gruppo	Descrizione
CSpreadBuffer	Classe buffer dello storico dello spread
CTimeBuffer	Classe buffer dello storico dei prezzi d'apertura
CTickVolumeBuffer	Classe buffer dello storico del volume tick
CRealVolumeBuffer	Classe buffer dello storico volume real
CDoubleBuffer	Classe base del buffer di dati di tipo double
COpenBuffer	Classe buffer dello storico dei prezzi barra
CHighBuffer	Classe buffer dei di High della barra
CLowBuffer	Classe buffer dei di Low della barra
CCloseBuffer	Classe buffer dei di Close della barra
CIndicatorBuffer	Classe buffer dell'indicatore Tecnico
CSeries	Classe di base per l'accesso ai dati timeseries
CPriceSeries	Classe di base per l'accesso ai dati dei prezzi
CIndicator	Classe base dell' indicatore tecnico
CIndicators	Collezione Indicatore tecnico e timeseries

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

CSpreadBuffer

CSpreadBuffer è una classe per l'accesso semplificato agli spread delle barre nello storico.

Descrizione

La classe CSpreadBuffer fornisce un accesso semplificato agli spread delle barre nello storico.

Dichiarazione

```
class CSpreadBuffer: public CArrayInt
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayInt
CSpreadBuffer

I Metodi della Classe per Gruppi

Attributi	
Size	Imposta la dimensione del buffer
Impostazioni	
SetSymbolPeriod	Imposta simbolo e di periodo
Accesso ai dati	
At	Ottiene l'elemento del buffer
Aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayInt

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Imposta la dimensione del buffer.

```
void  Size(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

SetSymbolPeriod

Imposta simboli e periodo.

```
void  SetSymbolPeriod(
   const string           symbol,      // simbolo(symbol)
   const ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Nuovo simbolo.

period

[in] Nuovo periodo (ENUM_TIMEFRAMES enumerazione).

At

Ottiene l'elemento buffer.

```
int  At(
   const int  index      // indice(index)
   ) const
```

Parametri

index

[in] Indice dell' elemento buffer.

Valore di ritorno

Elemento Buffer con l'indice specificato.

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CTimeBuffer

CTimeBuffer è una classe per l'accesso semplificato per gli orari di apertura delle barre nello storico.

Descrizione

La classe CTimeBuffer fornisce un accesso semplificato ai orari di apertura delle barre nello storico.

Dichiarazione

```
class CTimeBuffer: public CArrayLong
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayLong
      CTimeBuffer

I Metodi della Classe per Gruppi

Attributi	
Size	Imposta la dimensione del buffer
Impostazioni	
SetSymbolPeriod	Imposta simbolo e di periodo
Data Access Methods	
At	Ottiene l'elemento buffer per indice
Metodi di aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Imposta la dimensione del buffer.

```
void  Size(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

SetSymbolPeriod

Imposta simboli e periodo.

```
void  SetSymbolPeriod(
   const string           symbol,      // simbolo(symbol)
   const ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Nuovo simbolo.

period

[in] Nuovo periodo (ENUM_TIMEFRAMES enumerazione).

At

Ottiene l'elemento buffer.

```
long  At(
   const int  index      // indice(index)
   ) const
```

Parametri

index

[in] Indice dell' elemento buffer.

Valore di ritorno

Elemento Buffer con l'indice specificato.

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CTickVolumeBuffer

CTickVolumeBuffer è una classe per l'accesso semplificato ai volumi tick di barre nello storico.

Descrizione

La classe CTickVolumeBuffer fornisce un accesso semplificato ai ticks di barre nello storico.

Dichiarazione

```
class CTickVolumeBuffer: public CArrayLong
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayLong
      CTickVolumeBuffer

I Metodi della Classe per Gruppi

Attributi	
Size	Imposta la dimensione del buffer
Impostazioni	
SetSymbolPeriod	Imposta simbolo e di periodo
Data Access Methods	
At	Ottiene l'elemento buffer per indice
Metodi di aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Imposta la dimensione del buffer.

```
void  Size(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

SetSymbolPeriod

Imposta simboli e periodo.

```
void  SetSymbolPeriod(
   const string           symbol,      // simbolo(symbol)
   const ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Nuovo simbolo.

period

[in] Nuovo periodo (ENUM_TIMEFRAMES enumerazione).

At

Ottiene l'elemento buffer.

```
long  At(
   const int  index      // indice(index)
   ) const
```

Parametri

index

[in] Indice dell' elemento buffer.

Valore di ritorno

Elemento Buffer con l'indice specificato.

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CRealVolumeBuffer

CRealVolumeBuffer è una classe per l'accesso semplificato ai volumi reali di barre nello storico.

Descrizione

La Classe CTickVolumeBuffer offre un accesso semplificato ai volumi reali di barre nello storico.

Dichiarazione

```
class CRealVolumeBuffer: public CArrayLong
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayLong
      CRealVolumeBuffer

I Metodi della Classe per Gruppi

Attributi	
Size	Imposta la dimensione del buffer
Impostazioni	
SetSymbolPeriod	Imposta simbolo e di periodo
Accesso ai dati	
At	Ottiene l'elemento del buffer
Aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Imposta la dimensione del buffer.

```
void  Size(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

SetSymbolPeriod

Imposta simboli e periodo.

```
void  SetSymbolPeriod(
   const string           symbol,      // simbolo(symbol)
   const ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Nuovo simbolo.

period

[in] Nuovo periodo (ENUM_TIMEFRAMES enumerazione).

At

Ottiene l'elemento buffer.

```
long  At(
   const int  index      // indice(index)
   ) const
```

Parametri

index

[in] Indice dell' elemento buffer.

Valore di ritorno

Elemento Buffer con l'indice specificato.

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CDoubleBuffer

CDoubleBuffer è una classe di base per l'accesso semplificato al buffer di dati di tipo double.

Descrizione

La classe CDoubleBuffer fornisce un accesso semplificato ai dati del buffer di tipo double.

Dichiarazione

```
class CDoubleBuffer: public CArrayDouble
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayDouble
      CDoubleBuffer

Discendenti diretti

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

I Metodi della Classe per Gruppi

Attributi	
Size	Imposta la dimensione del buffer
Impostazioni	
SetSymbolPeriod	Imposta simbolo e di periodo
Accesso ai dati	
At	Ottiene l'elemento del buffer
Aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayDouble

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Imposta la dimensione del buffer.

```
void  Size(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

SetSymbolPeriod

Imposta simboli e periodo.

```
void  SetSymbolPeriod(
   const string           symbol,      // simbolo(symbol)
   const ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Nuovo simbolo.

period

[in] Nuovo periodo (ENUM_TIMEFRAMES enumerazione).

At

Ottiene l'elemento buffer.

```
double  At(
   const int  index      // indice(index)
   ) const
```

Parametri

index

[in] Indice dell' elemento buffer.

Valore di ritorno

Elemento Buffer con l'indice specificato.

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

COpenBuffer

COpenBuffer è una classe per l'accesso semplificato per i prezzi open nelle barre dello storico.

Descrizione

La Classe COpenBuffer offre un accesso semplificato ai prezzi open della barra nello storico.

Dichiarazione

```
class COpenBuffer: public CDoubleBuffer
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayDouble
CDoubleBuffer
COpenBuffer

I Metodi della Classe per Gruppi

Aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Metodi ereditati dalla classe CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CHighBuffer

CHighBuffer è una classe per l'accesso semplificato ai prezzi high della barra nello storico.

Descrizione

Classe CHighBuffer offre un accesso semplificato ai prezzi high nella barra dello storico.

Dichiarazione

```
class CHighBuffer: public CDoubleBuffer
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CHighBuffer

I Metodi della Classe per Gruppi

Aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Metodi ereditati dalla classe CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CLowBuffer

CLowBuffer è una classe per l'accesso semplificato ai prezzi low delle barre nello storico.

Descrizione

La classe CLowBuffer fornisce un accesso semplificato ai prezzi low delle barre nello storico.

Dichiarazione

```
class CLowBuffer: public CDoubleBuffer
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayDouble
CDoubleBuffer
CLowBuffer

I Metodi della Classe per Gruppi

Aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Metodi ereditati dalla classe CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CCloseBuffer

CCloseBuffer è una classe per l'accesso semplificato ai prezzi close della barra nello storico.

Descrizione

La Classe CCloseBuffer offre un accesso semplificato ai prezzi close della barra nello storico.

Dichiarazione

```
class CCloseBuffer: public CDoubleBuffer
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayDouble
CDoubleBuffer
CCloseBuffer

I Metodi della Classe per Gruppi

Aggiornamento dei dati	
virtual Refresh	Aggiorna il buffer
virtual RefreshCurrent	Aggiorna il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Metodi ereditati dalla classe CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Aggiorna il buffer.

```
virtual bool  Refresh()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna l'elemento corrente (zeresimo) del buffer.

```
virtual bool  RefreshCurrent()
```

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CIndicatorBuffer

CIndicatorBuffer è una classe per l'accesso semplificato ai dati del buffer dell'indicatore.

Descrizione

La Classe CIndicatorBuffer fornisce l'accesso semplificato al buffer dei dati dell' indicatore tecnico.

Dichiarazione

```
class CIndicatorBuffer: public CDoubleBuffer
```

Titolo

```
#include <Indicators\Indicator.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CIndicatorBuffer

I Metodi della Classe per Gruppi

Attributi	
Offset	Ottiene/Imposta l' offset del buffer
Name	Ottiene/Imposta il nome del buffer
Accesso ai dati	
At	Ottiene elemento del buffer
Aggiornamento dei dati	
Refresh	Aggiorna il buffer
RefreshCurrent	Aggiorna solo il valore corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Metodi ereditati dalla classe CDoubleBuffer

Size, At, SetSymbolPeriod

Offset

Ottiene offset del buffer.

```
int  Offset() const
```

Valore di ritorno

Offset del Buffer.

Offset

Imposta offset del buffer.

```
void  Offset(
   const int  offset      // offset
   )
```

Parametri

offset

[in] Nuovo offset del buffer.

Name

Ottiene il nome del buffer.

```
string  Name() const
```

Valore di ritorno

Nome del buffer.

Name

Imposta il nome del buffer.

```
void  Name(
   const string  name      // nome
   )
```

Parametri

name

[in] Nuovo nome del buffer.

At

Ottiene elemento buffer.

```
double  At(
   int  index      // indice
   ) const
```

Parametri

index

[in] Indice dell' elemento buffer.

Valore di ritorno

Elemento Buffer con l'indice specificato.

Refresh

Aggiorna l'intero buffer.

```
bool  Refresh(
   const int  handle,     // handle dell'indicatore
   const int  num         // numero del buffer
   )
```

Parametri

handle

[in] Handle dell'indicatore.

num

[in] Indice buffer dell'indicatore.

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

RefreshCurrent

Aggiorna il corrente (zeresimo) elemento del buffer.

```
bool  RefreshCurrent(
   const int  handle,     // handle dell'indicatore
   const int  num         // numero del buffer
   )
```

Parametri

handle

[in] Handle dell'indicatore.

num

[in] Numero buffer Indicatore.

Valore di ritorno

true - successo, false - non posso aggiornare il buffer.

CSeries

CSeries è una classe di base per l'accesso ai dati timeseries della Libreria Standard.

Descrizione

La Classe CSeries fornisce l'accesso semplificato a tutte le funzioni generali MQL5 API relative al lavoro con i dati serie per tutti i suoi discendenti (TimeSeries e classi indicatore).

Dichiarazione

```
class CSeries: public CArrayObj
```

Titolo

```
#include <Indicators\Series.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayObj
      CSeries

Discendenti diretti

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

I Metodi della Classe per Gruppi

Attributi	
Name	Ottiene il nome di timeseries o indicatore
BuffersTotal	Ottiene il numero di buffer di timeseries o indicatore
Timeframe	Ottiene la flag timeframe di timeseries o indicatore
Symbol	Ottiene il simbolo della timeseries o indicatore
Period	Ottiene il periodo di timeseries o indicatore
RefreshCurrent	Imposta/resetta il flag di aggiornamento dei dati correnti
Accesso ai dati	
virtual BufferResize	Imposta la grqandezza del buffer di timeseries o indicatore
Aggiornamento dei dati	
virtual Refresh	Aggiorna i dati di timeseries o indicatore
PeriodDescription	Trasforma ENUM_TIMEFRAMES in una stringa

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Name

Ottiene il nome della timeseries o indicatore.

```
string  Name() const
```

Valore di ritorno

Il nome della timeseries o indicatore.

BuffersTotal

Ottiene il numero di buffer di timeseries o indicatore.

```
int  BuffersTotal() const
```

Valore di ritorno

Il numero di buffer di timeseries o indicatore.

Nota

La timeseries ha solo un buffer.

Timeframe

Ottiene la flag del timeseries o indicatore.

```
int  Timeframe() const
```

Valore di ritorno

La flag timeframe di timeseries o indicatore.

Nota

E' generato come flag di visibilità oggetto.

Symbol

Ottiene il simbolo della timeseries o indicatore.

```
string  Symbol() const
```

Valore di ritorno

Il simbolo della timeseries o indicatore.

Period

Ottiene il periodo di timeseries o indicatore.

```
ENUM_TIMEFRAMES  Period() const
```

Valore di ritorno

Il periodo (valore dell'enumerazione ENUM_TIMEFRAMES) di timeseries o indicatore.

RefreshCurrent

Imposta un flag di aggiornamento costante dei i valori correnti di timeseries o indicatore.

```
string  RefreshCurrent(
   const bool  flag      // valore
   )
```

Parametri

flag

[in] Nuova flag.

Valore di ritorno

Nessuno.

BufferSize

Restituisce la quantità di dati disponibili nel buffer timeseries o buffer di indicatore.

```
int  BufferSize() const
```

Valore di ritorno

Quantità di dati disponibili in buffer timeseries o buffer di indicatore.

BufferResize

Imposta la dimensione del buffer di timeseries o indicatore.

```
virtual bool  BufferResize(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Nuova dimensione dei buffer.

Valore di ritorno

true - successo, altrimenti - false.

Nota

Tutte le timeseries o indicatore buffer hanno la stessa grandezza.

Refresh

Aggiorna i dati della timeseries o indicatore.

```
virtual void  Refresh(
   const int  flags      // flags
   )
```

Parametri

flags

[in] Timeframes da aggiornare (flag).

PeriodDescription

Ottiene la rappresentazione in formato stringa specificata enumerazione ENUM_TIMEFRAMES.

```
string  PeriodDescription(
   const int  val=0      // valore
   )
```

Parametri

val=0

[in] Valore da convertire.

Valore di ritorno

La rappresentazione in formato stringa specificata enumerazione ENUM_TIMEFRAMES.

Nota

Se il valore non è specificato o uguale a zero, timeframe dellei timeseries o indicatore vengono trasformati in una stringa.

CPriceSeries

CPriceSeries è una classe di base per l'accesso ai dati di prezzo.

Descrizione

La Classe CSeries fornisce l'accesso semplificato alle funzioni generali MQL5 API per lavorare con i dati sui prezzi a tutti i suoi discendenti.

Dichiarazione

```
class CPriceSeries: public CSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CPriceSeries

Discendenti diretti

CiClose, CiHigh, CiLow, CiOpen

I Metodi della Classe per Gruppi

Create	
virtual BufferResize	Imposta le dimensioni del buffer delle serie
Accesso ai dati	
virtual GetData	Ottiene l'elemento buffer delle serie specificato
Aggiornamento dei dati	
virtual Refresh	Aggiorna dati timeseries
Ricerca di valori estremi	
virtual MinIndex	Ottiene l'indice del valore minimo nell'intervallo specificato
virtual MinValue	Ottiene il valore minimo nell'intervallo specificato
virtual MaxIndex	Ottiene l'indice del valore massimo nell'intervallo specificato
virtual MaxValue	Ottiene il valore massimo nell'intervallo specificato

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

BufferResize

Imposta la dimensione del buffer serie.

```
virtual void  BufferResize(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

Valore di ritorno

true - successo, altrimenti - false.

GetData

Ottiene l'elemento buffer delle serie specificato.

```
double  GetData(
   const int  index      // indice(index)
   ) const
```

Parametri

index

[in] indice di un elemento buffer.

Valore di ritorno

L'elemento buffer delle serie, o EMPTY_VALUE.

Refresh

Aggiorna i dati timeseries

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // flags
   )
```

Parametri

flags=OBJ_ALL_PERIODS

[in] Timeframe flags.

MinIndex

Ottiene l'indice del valore minimo nell'intervallo specificato.

```
virtual int  MinIndex(
   const int  start,      // grandezza
   const int  count       // numero
   ) const
```

Parametri

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

Valore di ritorno

L'indice di valore minimo di un buffer serie nell'intervallo specificato, oppure -1.

MinValue

Ottiene il valore minimo nell'intervallo specificato.

```
virtual double  MinValue(
   const int    start,      // grandezza
   const int    count,      // numero
   int&         index       // riferimento
   ) const
```

Parametri

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

index

[out] riferimento alla variabile per l'immissione del valore dell' indice dell'elemento trovato.

Valore di ritorno

Il valore minimo del buffer serie nell'intervallo specificato, o EMPTY_VALUE.

Nota

L'indice dell'elemento trovato viene memorizzato per riferimento indice.

MaxIndex

Ottiene l'indice del valore massimo nell'intervallo specificato.

```
virtual int  MaxIndex(
   const int  start,      // indice(index)
   const int  count       // numero
   ) const
```

Parametri

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

Valore di ritorno

L'indice del valore massimo del buffer serie nell'intervallo specificato, oppure -1.

MaxValue

Ottiene il valore massimo nell'intervallo specificato.

```
virtual double  MaxValue(
   const int    start,      // grandezza
   const int    count,      // ammontare
   int&         index       // riferimento
   ) const
```

Parametri

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

index

[out] riferimento alla variabile per l'immissione del valore dell' indice dell'elemento trovato.

Valore di ritorno

Il valore massimo di un buffer serie nell'intervallo specificato, o EMPTY_VALUE.

Nota

L'indice dell'elemento trovato viene memorizzato per riferimento indice.

CIndicator

CIndicator è una classe base per le classi indicatore tecnico della libreria standard MQL5.

Descrizione

La classe CIndicator fornisce l'accesso semplificato per tutti i suoi discendenti a funzioni generali di indicatore tecnico MQL5 API.

Dichiarazione

```
class CIndicator: public CSeries
```

Titolo

```
#include <Indicators\Indicator.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator

Discendenti diretti

CiAC, CiAD, CiADX, CiADXWilder, CiAlligator, CiAMA, CiAO, CiATR, CiBands, CiBearsPower, CiBullsPower, CiBWMFI, CiCCI, CiChaikin, CiCustom, CiDEMA, CiDeMarker, CiEnvelopes, CiForce, CiFractals, CiFrAMA, CiGator, CiIchimoku, CiMA, CiMACD, CiMFI, CiMomentum, CiOBV, CiOsMA, CiRSI, CiRVI, CiSAR, CiStdDev, CiStochastic, CiTEMA, CiTriX, CiVIDyA, CiVolumes, CiWPR

I Metodi della Classe per Gruppi

Attributi	
Handle	Ottiene l'handle dell'indicatore.
Status	Ottiene lo stato dell'indicatore.
FullRelease	Imposta una flag per rilasciare l'handle.
Creazione	
Create	Crea l'indicatore
BufferResize	Imposta le nuove dimensioni del buffer
Accesso ai dati	
GetData	Copia i dati dal buffer indicatore
Metodi di aggiornamento dei dati	

Attributi	
Refresh	Aggiorna i dati dell' indicatore
Ricerca di valori Min/Max	
Minimum	Ottiene l'indice del valore minimo in un intervallo specificato.
MinValue	Ottiene il valore minimo in un intervallo specificato.
Maximum	Ottiene l'indice del valore massimo in un intervallo specificato.
MaxValue	Ottiene il valore massimo in un intervallo specificato.
Conversione dell' Enumerazione	
MethodDescription	Converte ENUM_MA_METHOD in una stringa
PriceDescription	Converte ENUM_APPLIED_PRICE in una stringa
VolumeDescription	Converte ENUM_APPLIED_VOLUME in una stringa
Lavorare con il chart	
AddToChart	Aggiunge un indicatore al grafico
DeleteFromChart	Elimina un indicatore dal grafico

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Handle

Ottiene l'handle dell'indicatore.

```
int  Handle() const
```

Valore di ritorno

Handle dell'indicatore.

Status

Ottiene lo stato dell'indicatore.

```
string  Status() const
```

Valore di ritorno

Lo stato di creazione dell'indicatore.

FullRelease

Imposta una flag per rilasciare l'handle.

```
void  FullRelease(
   const bool  flag=true      // flag
   )
```

Parametri

flag

[in] Nuovo valore della flag di rilascio handle.

Create

Crea l'indicatore del tipo specificato con i parametri specificati.

```
bool  Create(
   const string           symbol,          // simbolo
   const ENUM_TIMEFRAMES  period,          // periodo
   const ENUM_INDICATOR   type,            // tipo
   const int              num_params,      // numero di parametri
   const MqlParam&        params[]         // array di parametri
   )
```

Parametri

symbol

[in] Simbolo dell'indicatore.

period

[in] Timefframe dell'indicatore (enumerazione ENUM_TIMEFRAMES).

type

[in] Tipo dell'indicatore (enumerazione ENUM_INDICATOR).

num_params

[in] Numero di parametri dell'indicatore.

params

[in] Riferimento ai parametri dell'array dell'indicatore.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

BufferResize

Imposta la grandezza dei buffer indicatore.

```
virtual bool  BufferResize(
   const int  size      // grandezza(size)
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

Valore di ritorno

true - successo, altrimenti - false.

Nota

Tutti gli indicatori del buffer hanno la stessa grandezza.

GetData

Ottiene l'elemento specificato del buffer specificato dell'indicatore. Refresh() dovrebbe essere chiamato per lavorare con i dati recenti prima di utilizzare il metodo.

```
double  GetData(
   const int  buffer_num,     // numero buffer
   const int  index           // indice(index)
   ) const
```

Parametri

buffer_num

[in] Numero buffer Indicatore.

index

[in] Indice dell'elemento buffer indicatore.

Valore di ritorno

valore - successo, EMPTY_VALUE - non può ricevere i dati.

GetData

Ottiene i dati dal buffer dell'indicatore per posizione e numero di partenza.

```
int  GetData(
   const int      start_pos,      // posizione
   const int      count,          // numero
   const int      buffer_num,     // numero buffer
   double&        buffer[]        // array
   ) const
```

Parametri

start_pos

[in] La posizione di partenza del buffer indicatore .

count

[in] Numero di elementi del buffer indicatore.

buffer_num

[in] Numero di buffer dell'indicatore.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

Numero dei valori dell' indicatore ricevuti dal buffer indicatore specificato - successo, altrimenti -1.

GetData

Ottiene i dati dal buffer indicatore per tempo di inizio e numero.

```
int  GetData(
   const datetime  start_time,     // orario d'inizio
   const int       count,          // ammontare
   const int       buffer_num,     // numero buffer
   double&         buffer[]        // array
   ) const
```

Parametri

start_time

[in] Elemento indicatore buffer ora di inizio.

count

[in] Numero di elementi del buffer indicatore.

buffer_num

[in] Numero di buffer dell'indicatore.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

Numero dei valori degli indicatori ricevuti dal buffer specificato, altrimenti -1.

GetData

Ottiene i dati dal buffer indicatore per orario di inizio e di fine.

```
int  GetData(
   const datetime  start_time,     // orario d'inizio
   const datetime  stop_time,      // orario di fine
   const int       buffer_num,     // numero di buffer
   double&         buffer[]        // array
   ) const
```

Parametri

start_time

[in] Orario dell'elemento iniziale del buffer indicatore.

stop_time

[in] Orario dell'elemento finale del buffer indicatore.

buffer_num

[in] Numero di buffer dell'indicatore.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

Numero dei valori degli indicatori ricevuti dal buffer specificato - successo, altrimenti -1.

Refresh

Aggiorna i dati degli indicatori. E' consigliabile chiamare il metodo prima di utilizzare GetData().

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // flags
   )
```

Parametri

flags=OBJ_ALL_PERIODS

[in] Flags di aggiornamento timeframe.

Minimum

Restituisce l'indice dell'elemento minima del buffer specificato in un intervallo specificato.

```
int  Minimum(
   const int  buffer_num,     // numero buffer
   const int  start,          // indice iniziale
   const int  count           // numero
   ) const
```

Parametri

buffer_num

[in] Numero Buffer in cui cercare il valore.

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

Valore di ritorno

Indice dell'elemento minima del buffer specificato in un intervallo specificato.

MinValue

Restituisce il valore dell'elemento minimo del buffer specificato in un intervallo specificato.

```
double  MinValue(
   const int   buffer_num,     // numero buffer
   const int   start,          // indice iniziale
   const int   count,          // numero
   int&        index           // riferimento
   ) const
```

Parametri

buffer_num

[in] Numero Buffer in cui cercare il valore.

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

index

[out] Riferimento alla variabile per memorizzare il valore dell'indice dell'elemento trovato.

Valore di ritorno

Il valore dell'elemento minima del buffer specificato in un intervallo specificato.

Nota

L'indice dell'elemento trovato viene memorizzato per riferimento indice.

Maximum

Restituisce l'indice dell'elemento massimo del buffer specificato in un intervallo specificato.

```
int  Maximum(
   const int  buffer_num,     // numero buffer
   const int  start,          // indice iniziale
   const int  count           // numero
   ) const
```

Parametri

buffer_num

[in] Numero Buffer in cui cercare il valore.

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

Valore di ritorno

Indice dell'elemento massimo del buffer specificato in un intervallo specificato.

MaxValue

Restituisce il valore dell'elemento massimo del buffer specificato in un intervallo specificato.

```
double  MaxValue(
   const int   buffer_num,     // numero buffer
   const int   start,          // indice iniziale
   const int   count,          // numero
   int&        index           // riferimento
   ) const
```

Parametri

buffer_num

[in] Numero Buffer in cui cercare il valore.

start

[In] Indice iniziale di range di ricerca .

count

[in] Grandezza del reange di ricerca (numero di elementi).

index

[out] Riferimento alla variabile per memorizzare il valore dell'indice dell'elemento trovato.

Valore di ritorno

Il valore dell'elemento massimo del buffer specificato in un intervallo specificato.

Nota

L'indice dell' elemento massimo del buffer è memorizzato per riferimento indice.

MethodDescription

Converte ENUM_MA_METHOD valore di enumerazione in una stringa.

```
string  MethodDescription(
   const int  val      // valore
   ) const
```

Parametri

val

[in] Valore della conversione.

Valore di ritorno

La stringa corrispondente al valore dell' enumerazione ENUM_MA_METHOD.

PriceDescription

Converte il valore dell' enumerazione ENUM_APPLIED_PRICE in una stringa.

```
string  PriceDescription(
   const int  val      // valore
   ) const
```

Parametri

val

[in] Valore della conversione.

Valore di ritorno

La stringa corrispondente al valore dell' enumerazioneENUM_APPLIED_PRICE .

VolumeDescription

Converte il valore dell' enumerazione ENUM_APPLIED_VOLUME in una stringa.

```
string  VolumeDescription(
   const int  val      // valore
   ) const
```

Parametri

val

[In] Valore della conversione.

Valore di ritorno

La stringa corrispondente al valore dell' enumerazione ENUM_APPLIED_VOLUME.

AddToChart

Aggiunge l'indicatore al chart.

```
bool  AddToChart(
   const long  chart,      // chart ID
   const int  subwin       // indice sottofinestra
   )
```

Parametri

chart

[in] Chart ID.

subwin

[in] Indice sottofinestra Chart.

Valore di ritorno

true - successo, false - non può aggiungere l'indicatore al grafico.

DeleteFromChart

Elimina l'indicatore dal chart.

```
bool  DeleteFromChart(
   const long  chart,      // chart ID
   const int  subwin       // indice sottofinestra
   )
```

Parametri

chart

[in] Chart ID.

subwin

[in] Indice sottofinestra Chart.

Valore di ritorno

true - successo, false - non può rimuovere l'indicatore dal chart.

BarsCalculated

Restituisce la quantità di dati calcolati per l'indicatore.

```
int  BarsCalculated() const;
```

Valore di ritorno

Restituisce la quantità di dati calcolati nel buffer indicatore, o -1 in caso di errore (dati non ancora calcolati).

CIndicators

CIndicators è una classe per la raccolta di istanze di timeseries e classi di indicatori tecnici.

Descrizione

La Classe CIndicators fornisce la creazione delle istanze di classe di indicatori tecnici, la loro archiviazione e la gestione (la sincronizzazione dei dati, handle e management della memoria).

Dichiarazione

```
class CIndicators: public CArrayObj
```

Titolo

```
#include <Indicators\Indicators.mqh>
```

I Metodi della Classe per Gruppi

Create	
Create	Crea un indicatore
Aggiornamento dei dati	
Refresh	Aggiorna dati Indicatore

Create

Crea l'indicatore del tipo specificato con i parametri specificati.

```
CIndicator*  Create(
   const string           symbol,      // nome simbolo
   const ENUM_TIMEFRAMES  period,      // periodo
   const ENUM_INDICATOR   type,        // tipo
   const int              count,       // numero di parametri
   const MqlParam&        params       // array parametri
   )
```

Parametri

symbol

[in] Nome simbolo dell'indicatore.

period

[in] Indicatore timeframe (ENUM_TIMEFRAMES valore dell' enumerazione).

type

[in] Tipo dell'indicatore (valore enumerazione ENUM_INDICATOR).

count

[in] Numero di parametri per l'indicatore.

params

[in] Riferimento ai parametri dell'array dell'indicatore.

Valore di ritorno

Riferimento all'indicatore creato - successo, NULL - non si può creare l'indicatore.

Refresh

Aggiornamenti dei dati per tutte le timeseries e gli indicatori tecnici nella raccolta.

```
int  Refresh()
```

Valore di ritorno

Flags aggiornamento timeframes (formate da flags di visibilità oggetto).

Classi Timeseries

Questo gruppo di capitoli contiene dettagli tecnici di classi timeseries della Libreria Atandard MQL5 e le descrizioni di tutte le sue componenti principali.

Classe	Descrizione
CiSpread	Fornisce un accesso agli spread dei dati storici
CiTime	Fornisce un accesso agli orari open delle barre nello storico
CiTickVolume	Fornisce un accesso ai volumi tick delle barre nello storico
CiRealVolume	Fornisce un accesso ai volumi real delle barre nello storico
CiOpen	Fornisce un accesso per ai prezzi open delle barre nello storico
CiHigh	Fornisce un accesso ai prezzi high delle barre nello storico
CiLow	Fornisce un accesso ai prezzi low delle barre nello storico
CiClose	Fornisce un accesso ai prezzi close delle barre nello storico

CiSpread

CiSpread è una classe progettato per l'accesso agli spread delle barre nello storico.

Descrizione

La Classe CiSpread fornisce un accesso agli spread dei dati storici.

Dichiarazione

```
class CiSpread: public CSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CiSpread

I Metodi della Classe per Gruppi

Create	
Create	Crea una timeseries
BufferResize	Imposta la grandezza del buffer serie
Accesso ai dati	
GetData	Ottiene i dati della serie
Aggiornamento dei dati	
Refresh	Aggiorna i dati della serie

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Crea un timeseries con i parametri specificati per l'accesso allo storico degli spread.

```
bool  Create(
   string           symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo timeseries.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare le timeseries.

BufferResize

Imposta la grandezza delle serie buffer.

```
virtual void  BufferResize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

GetData

Ottiene l'elemento specificato di timeseries buffer.

```
int  GetData(
   int  index      // indice
   ) const
```

Parametri

index

[in] indice dell'elemento del buffer.

Valore di ritorno

L'elemento del buffer timeseries, oppure 0.

GetData

Ottiene l'elemento di timeseries per posizione e il numero di partenza.

```
int  GetData(
   int   start_pos,     // posizione
   int   count,         // numero
   int&  buffer         // array
   ) const
```

Parametri

start_pos

[in] La posizione di partenza di un buffer timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento all'array per la memorizzazione dei dati.

Valore di ritorno

>= 0 - successo, -1 - non può ricevere i dati.

GetData

Ottiene dati da un buffer timeseries per orario e numero d'inizio.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   int       count,          // numero
   int&      buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell'elemento iniziale del buffer di timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati da un buffer timeseries per orari di inizio e fine.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   datetime  stop_time,      // orario di fine
   int&      buffer          // array
   ) const
```

Parametri

start_time

[in] Orario d'inizio di un elemento buffer di una timeseries.

stop_time

[in] Orario di fine di un elemento buffer di una timeseries.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

>= 0 - successo, -1 - non può ricevere i dati.

Refresh

Aggiorna i dati di timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parametri

flags

[in] Timeframe flags.

CiTime

CiTime è una classe progettata per l'accesso agli orari open delle barre nello storico.

Descrizione

La Classe CiTime fornisce un accesso agli orari open delle barre nello storico.

Dichiarazione

```
class CiTime: public CSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CiTime

I Metodi della Classe per Gruppi

Create	
Create	Crea una timeseries
BufferResize	Imposta le dimensioni del buffer timeseries
Accesso ai dati	
GetData	Ottiene i dati timeseries
Aggiornamento dei dati	
Refresh	Aggiorna i dati timeseries

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Crea una timeseries con i parametri specificati per l'accesso agli orari di apertura, delle barre nello storico.

```
bool  Create(
   string           symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo timeseries.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare le timeseries.

BufferResize

Imposta le dimensioni del buffer delle serie.

```
virtual void  BufferResize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

GetData

Ottiene l'elemento specificato di timeseries buffer.

```
datetime  GetData(
   int  index      // indice
   ) const
```

Parametri

index

[in] indice dell'elemento del buffer.

Valore di ritorno

L'elemento del buffer timeseries, oppure 0.

GetData

Ottiene i dati da un buffer timeseries per posizione e numero di partenza.

```
int  GetData(
   int   start_pos,     // posizione
   int   count,         // numero
   long& buffer         // array
   ) const
```

Parametri

start_pos

[in] Posizione di partenza del timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati dal buffer timeseries per orario e numero di partenza.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   int       count,          // numero
   long&     buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati da un buffer timeseries per orario di inizio e fine.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   datetime  stop_time,      // orario di fine
   long&     buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

stop_time

[in] Oraro di un elemento finale del buffer timeseries.

buffer

[in] Riferimento all'array per memorizzare i dati

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

Refresh

Aggiorna i dati di timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parametri

flags

[in] Timeframe flags.

CiTickVolume

CiTickVolume è una classe progettata per l'accesso ai volumi tick delle barre nello storico.

Descrizione

Classe CiTickVolume fornisce un accesso ai volumi tick delle barre nello storico.

Dichiarazione

```
class CiTickVolume: public CSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CiTickVolume

I Metodi della Classe per Gruppi

Create	
Create	Crea una timeseries
BufferResize	Imposta le dimensioni del buffer
Accesso ai dati	
GetData	Ottiene i dati della serie
Aggiornamento dei dati	
Refresh	Aggiorna i dati della serie

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Crea una timeseries con i parametri specificati per l'accesso ai volumi tick delle barre nello storico.

```
bool  Create(
   string           symbol,     // simbolo(symbol)
   ENUM_TIMEFRAMES  period      // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo Timeseires.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare una timeseries.

BufferResize

Imposta dimensione delle serie buffer.

```
virtual void  BufferResize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

GetData

Ottiene l'elemento timeseries del buffer specificato.

```
datetime  GetData(
   int  index      // indice
   ) const
```

Parametri

index

[in] indice dell'elemento del buffer.

Valore di ritorno

L'elemento del buffer timeseries, oppure 0.

GetData

Ottiene i dati dal buffer timeseries per posizione e numero iniziale.

```
int  GetData(
   int   start_pos,     // posizione
   int   count,         // numero
   long& buffer         // array
   ) const
```

Parametri

start_pos

[in] La posizione di partenza di un buffer timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento all'array per la memorizzazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati da un buffer timeseries per orario e numero di partenza.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   int       count,          // numero
   long&     buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell'elemento iniziale del buffer timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento all'array per memorizzare i dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati da un buffer timeseries per orario di inizio e fine.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   datetime  stop_time,      // orario di fine
   long&     buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell'elemento iniziale del buffer timeseries.

stop_time

[in] Orario dell'elemento finale del buffer timeseries.

buffer

[in] Riferimento all'array per memorizzare i dati

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

Refresh

Aggiorna i dati di timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parametri

flags

[in] Timeframe flags.

CiRealVolume

CiRealVolume è una classe progettata per l'accesso ai volumi reali (real volumes) delle barre nello storico.

Descrizione

La Classe CiRealVolume fornisce un accesso ai volumi reali delle barre nellp storico.

Dichiarazione

```
class CiRealVolume: public CSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CiRealVolume

I Metodi della Classe per Gruppi

Create	
Create	Crea una serie
BufferResize	Imposta la dimensione del buffer
Accesso ai dati	
GetData	Ottiene i dati della serie
Aggiornamento dei dati	
Refresh	Aggiorna i dati della serie

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Crea un timeseries con i parametri specificati per l'accesso ai volumi reali delle barre nello storico.

```
bool  Create(
   string           symbol,     // simbolo(symbol)
   ENUM_TIMEFRAMES  period      // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo timeseries.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare una timeseries.

BufferResize

Imposta la grandezza del buffer serie.

```
virtual void  BufferResize(
   int  size      // grandezza
   )
```

Parametri

size

[in] Dimensione nuova del buffer.

GetData

Ottiene l'elemento buffer delle serie specificato.

```
datetime  GetData(
   int  index      // indice
   ) const
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento buffer della serie, oppure 0.

GetData

Ottiene i dati dal buffer timeseries per posizione e numero di partenza.

```
int  GetData(
   int   start_pos,     // posizione
   int   count,         // numero
   long& buffer         // array
   ) const
```

Parametri

start_pos

[in] Posizione di partenza del timeseries buffer.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati dal buffer timeseries per orario e numero di partenza.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   int       count,          // numero
   long&     buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell'elemento iniziale del buffer timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene l'elemento delle timeseries per orario di avvio e stop.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   datetime  stop_time,      // orario di fine
   long&     buffer          // target array
   ) const
```

Parametri

start_time

[in] Orario d'inizio.

stop_time

[in] Orario di stop.

buffer

[in] Riferimento all'array target, per i dati

Valore di ritorno

>=0 in caso di successo, -1 in caso di errore.

Refresh

Aggiorna i dati di timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parametri

flags

[in] Timeframe flags.

CiOpen

CiOpen è una classe progettato per l'accesso ai prezzi open delle barre nello storico.

Descrizione

La Classe CiOpen fornisce un accesso a prezzi open delle barre nello storico.

Dichiarazione

```
class CiOpen: public CPriceSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiOpen

I Metodi della Classe per Gruppi

Create	
Create	Crea una serie
Accesso ai dati	
GetData	Ottiene i dati della serie

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Crea una timeseries con i parametri specificati per l'accesso ai prezzi open delle barre nello storico.

```
bool  Create(
   string           symbol,     // simbolo(symbol)
   ENUM_TIMEFRAMES  period      // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo timeseries.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare una timeseries.

GetData

Ottiene l'elemento di timeseries per posizione e il numero di partenza.

```
int  GetData(
   int    start_pos,      // posizione di partenza
   int    count,          // numero
   double& buffer         // array
   ) const
```

Parametri

start_pos

[in] Posizione di partenza del timeseries buffer.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati dal buffer timeseries per orario e numero di partenza.

```
int  GetData(
   datetime  start_time,      // orario d'inizio
   int       count,           // numero
   double&     buffer         // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati dal buffer timeseries per orari di inizio e fine.

```
int  GetData(
   datetime  start_time,      // orario d'inizio
   datetime  stop_time,       // orario di fine
   double&   buffer           // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

stop_time

[in] Orario dell' ultimo elemento di un buffer di timeseries .

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

CiHigh

CiHigh è una classe progettato per l'accesso ai prezzi high delle barre nello storico.

Descrizione

La Classe CiHigh fornisce un accesso a prezzi high delle barre nello storico.

Dichiarazione

```
class CiHigh: public CPriceSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiHigh

I Metodi della Classe per Gruppi

Create	
Create	Crea una serie
Accesso ai dati	
GetData	Ottiene i dati della serie

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Crea un timeseries con i parametri specificati per l'accesso ai prezzi high delle barre nello storico.

```
bool  Create(
   string           symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo timeseries.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare una timeseries.

GetData

Ottiene i dati dal buffer timeseries per posizione e numero iniziale.

```
int  GetData(
   int    start_pos,      // posizione
   int    count,          // numero
   double& buffer         // array
   ) const
```

Parametri

start_pos

[in] La posizione di partenza di un buffer timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati dal buffer timeseries per orario e numero di partenza.

```
int  GetData(
   datetime  start_time,      // orario d'inizio
   int       count,           // numero
   double&     buffer         // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati dal buffer timeseries per orario d'inizio e fine.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   datetime  stop_time,      // orario di fine
   double&   buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

stop_time

[in] Orario dell' ultimo elemento di un buffer di timeseries .

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

CiLow

CiLow è una classe progettato per l'accesso ai prezzi low delle barre nello storico.

Descrizione

La Classe CiLow fornisce un accesso a prezzi low delle barre nello storico.

Dichiarazione

```
class CiLow: public CPriceSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiLow

I Metodi della Classe per Gruppi

Create	
Create	Crea una serie
Accesso ai dati	
GetData	Ottiene un serie di dati

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Crea un timeseries con i parametri specificati per l'accesso ai prezzi low delle barre nello storico.

```
bool  Create(
   string           symbol,     // simbolo(symbol)
   ENUM_TIMEFRAMES  period      // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo timeseries.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare le timeseries.

GetData

Ottiene l'elemento di timeseries per posizione e il numero di partenza.

```
int  GetData(
   int    start_pos,      // posizione
   int    count,          // numero
   double& buffer         // array
   ) const
```

Parametri

start_pos

[in] La posizione di partenza di un buffer timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati da un buffer timeseries per orario e numero di partenza.

```
int  GetData(
   datetime  start_time,      // orario d'inizio
   int       count,           // numero
   double&     buffer         // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati dal buffer timeseries per orario d'inizio e fine.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   datetime  stop_time,      // orario di fine
   double&   buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

stop_time

[in] Orario dell' ultimo elemento di un buffer di timeseries .

buffer

[in] Riferimento per l'array di archiviazione dati

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

CiClose

CiClose è una classe designata per l'accesso ai prezzi close delle barre nello storico.

Descrizione

Classe CiClose fornisce un accesso ai prezzi close delle barre nello storico.

Dichiarazione

```
class CiClose: public CPriceSeries
```

Titolo

```
#include <Indicators\TimeSeries.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiClose

I Metodi della Classe per Gruppi

Create	
Create	Crea una serie
Accesso ai dati	
GetData	Ottiene i dati della serie

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Crea una timeseries con i parametri specificati per l'accesso ai prezzi close delle barre nello storico.

```
bool  Create(
   string           symbol,     // simbolo(symbol)
   ENUM_TIMEFRAMES  period      // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo timeseries.

period

[in] Timeseries timeframe (valore dell'enumerazioneENUM_TIMEFRAMES).

Valore di ritorno

true - successo, false - non può creare le timeseries.

GetData

Ottiene i dati da un buffer timeseries per posizione e numero di partenza.

```
int  GetData(
   int    start_pos,      // posizione
   int    count,          // numero
   double& buffer         // array
   ) const
```

Parametri

start_pos

[in] La posizione di partenza di un buffer timeseries.

count

[in] Numero di elementi buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati da un buffer timeseries per orario e numero d'inizio.

```
int  GetData(
   datetime  start_time,      // orario d'inizio
   int       count,           // numero
   double&     buffer         // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

count

[in] Numero di un elemento di buffer timeseries.

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

GetData

Ottiene i dati da un buffer timeseries per orari di inizio e fine.

```
int  GetData(
   datetime  start_time,     // orario d'inizio
   datetime  stop_time,      // orario di fine
   double&   buffer          // array
   ) const
```

Parametri

start_time

[in] Orario dell' elemento iniziale di un buffer timeseries .

stop_time

[in] Orario dell' ultimo elemento di un buffer di timeseries .

buffer

[in] Riferimento per l'array di archiviazione dei dati.

Valore di ritorno

>=0 - successo, -1 - non si possono ricevere i dati.

Main and Auxiliary Classes of Technical Indicators and Timeseries

Questo gruppo di capitoli contiene i dettagli tecnici delle principali e ausiliari classi di indicatori tecnici e timeseries, così come le descrizioni dei componenti appropriati della Libreria Standard MQL5.

Classe/gruppo	Descrizione
CiADX	Average Directional Index
CiADXWilder	Average Directional Index by Welles Wilder
CiBands	Bollinger Bands®
CiEnvelopes	Envelopes
CiIchimoku	Ichimoku Kinko Hyo
CiMA	Moving Average
CiSAR	Parabolic Stop And Reverse System
CiStdDev	Standard Deviation
CiDEMA	Double Exponential Moving Average
CiTEMA	Triple Exponential Moving Average
CiFrAMA	Fractal Adaptive Moving Average
CiAMA	Adaptive Moving Average
CiVIDyA	Variable Index Dynamic Average

CiADX

CiADX è una classe destinata per utilizzo dell'indicatore tecnico Average Directional Index.

Descrizione

classe CiADX fornisce la creazione, la configurazione e l'accesso ai dati dell'indicatore Average Directional Index.

Dichiarazione

```
class CiADX: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADX

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce l'elemento di buffer della linea principale
Plus	Restituisce l'elemento di buffer della linea +DI
Minus	Restituisce l'elemento di buffer della linea -DI
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento di buffer della linea principale per indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento di buffer della linea principale .

Valore di ritorno

Elemento della linea principale del buffer per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Plus

Restituisce l'elemento del buffer della linea +DI dall'indice specificato.

```
double  Plus(
   int  index      // indice
   )
```

Parametri

index

[In] Elemento indice del buffer della linea +DI.

Valore di ritorno

L'elemento di buffer della linea + DI per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Minus

Restituisce l'elemento di buffer della linea -DI per indice specificato.

```
double  Minus(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento buffer della linea -DI .

Valore di ritorno

L'elemento di buffer della linea -DI per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_ADX per CiADX).

CiADXWilder

CiADXWilder è una classe destinata per utilizzo dell'indicatore tecnico Average Directional Index by Welles Wilder.

Descrizione

Classe CiADXWilder prevede la creazione, la configurazione e l'accesso ai dati del Average Directional Index by Welles Wilder.

Dichiarazione

```
class CiADXWilder: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADXWilder

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati buffer della linea principale
Plus	Restituisce i dati del buffer della linea +DI
Minus	Restituisce i dati del buffer della linea -DI
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento di buffer della linea principale per indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento di buffer della linea principale .

Valore di ritorno

Elemento della linea principale del buffer per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Plus

Restituisce l'elemento del buffer della linea +DI dall'indice specificato.

```
double  Plus(
   int  index      // indice
   )
```

Parametri

index

[In] Elemento indice del buffer della linea +DI.

Valore di ritorno

L'elemento di buffer della linea + DI per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Minus

Restituisce l'elemento di buffer della linea -DI per indice specificato.

```
double  Minus(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento buffer della linea -DI .

Valore di ritorno

L'elemento di buffer della linea -DI per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_ADXW per CiADXWilder).

CiBands

CiBands è una classe destinata all' utilizzo dell' Indicatore tecnico Bollinger Bands®.

Descrizione

La Classe CiAMA prevede la creazione, la configurazione e l'accesso ai dati dell' indicatore Adaptive Moving Average.

Dichiarazione

```
class CiBands: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBands

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
MaShift	Restituisce lo slittamento orizzontale
Deviation	Restituisce la deviazione
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Base	Restituisce l'elemento del buffer della linea "base" (base line)
Upper	Restituisce l'elemento del buffer della linea "upper" (upper line)
Lower	Restituisce l'elemento del buffer della riga inferiore
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

MaShift

Restituisce lo slittamento orizzontale.

```
int  MaShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

Deviation

Restituisce la deviazione.

```
double  Deviation() const
```

Valore di ritorno

Restituisce la deviazione, definita alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,         // simbolo
   ENUM_TIMEFRAMES  period,         // periodo
   int              ma_period,      // periodo medio
   int              ma_shift,       // slittamento
   double           deviation,      // deviazione
   int              applied         // prezzo applicato, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ma_shift

[in] Slittamento orizzontale dell'indicatore.

deviation

[in] Deviazione.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Base

Restituisce l'elemento del buffer della linea di base per l'indice specificato.

```
double  Base(
   int  index      // indice
   )
```

Parametri

index

[in] Indice della linea Base dell'elemento del buffer.

Valore di ritorno

L'elemento del buffer della linea base dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Upper

Restituisce l'elemento del buffer della linea superiore dall'indice specificato.

```
double  Upper(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento buffer della linea superiore.

Valore di ritorno

L'elemento del buffer della linea superiore dell' indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Lower

Restituisce l'elemento del buffer della linea inferiore dell' indice specificato.

```
double  Lower(
   int  index      // indice
   )
```

Parametri

index

[in] Elemento dell'indice del buffer Lower Line.

Valore di ritorno

L'elemento del buffer della riga inferiore dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_BANDS per CiBands).

CiEnvelopes

CiEnvelopes è una classe destinata all'utilizzo dell' indicatore tecnico Envelopes.

Descrizione

La Classe CiEnvelopes prevede la creazione, la configurazione e l'accesso ai dati dell' indicatore Envelopes.

Dichiarazione

```
class CiEnvelopes: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiEnvelopes

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
MaShift	Restituisce lo slittamento orizzontale
MaMethod	Restituisce il metodo di calcolo della media
Deviation	Restituisce la deviazione
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Upper	Restituisce i dati del buffer della linea superiore
Lower	Restituisce i dati del buffer della linea più bassa
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

MaShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  MaShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo di calcolo della media, definita alla creazione dell'indicatore (valore dell' enumerazione ENUM_MA_METHOD).

Deviation

Restituisce il valore di scostamento(deviation).

```
double  Deviation() const
```

Valore di ritorno

Restituisce il valore di scostamento, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period,     // periodo medio
   int              ma_shift,      // slittamento
   ENUM_MA_METHOD   ma_method,     // metodo della media
   int              applied,       // tipo di prezzo, handle
   double           deviation      // deviazione
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ma_shift

[in] Slittamento dell'asse prezzi.

ma_method

[in] Metodo di media (ENUM_MA_METHOD, valore dell'enumerazione).

applied

[in] Oggetto (tipo di prezzo o handle) da applicare.

deviation

[in] Deviazione.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Upper

Restituisce l'elemento del buffer della linea superiore dall'indice specificato.

```
double  Upper(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento buffer della linea superiore.

Valore di ritorno

L'elemento di buffer della linea superiore dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Lower

Restituisce l'elemento del buffer della linea inferiore dell' indice specificato.

```
double  Lower(
   int  index      // indice
   )
```

Parametri

index

[in] Elemento dell'indice del buffer Lower Line.

Valore di ritorno

L'elemento del buffer della riga inferiore dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_ENVELOPES per CiEnvelopes).

CiIchimoku

CiIchimoku è una classe destinata all' utilizzo dell'indicatore tecnico Ichimoku Kinko Hyo.

Descrizione

La Classe CiIchimoku prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Ichimoku Kinko Hyo.

Dichiarazione

```
class CiIchimoku: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiIchimoku

I Metodi della Classe per Gruppi

Attributi	
TenkanSenPeriod	Restituisce il periodo TenkanSen
KijunSenPeriod	Restituisce il periodo KijunSen
SenkouSpanBPeriod	Restituisce il periodo SenkouSpanB
Create	
Create	Crea l'indicatore
Accesso ai dati	
TenkanSen	Restituisce l'elemento di buffer della linea TenkanSen
KijunSen	Restituisce l'elemento di buffer della linea KijunSen
SenkouSpanA	Restituisce l'elemento di buffer della linea SenkouSpanA
SenkouSpanB	Restituisce l'elemento di buffer della linea SenkouSpanB
ChinkouSpan	Restituisce l'elemento di buffer della linea ChikouSpan
Input/output	

Attributi	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

TenkanSenPeriod

Restituisce il periodo TenkanSen.

```
int  TenkanSenPeriod() const
```

Valore di ritorno

Restituisce il periodo TenkanSen, definito alla creazione dell'indicatore.

KijunSenPeriod

Restituisce il periodo KijunSen.

```
int  KijunSenPeriod() const
```

Valore di ritorno

Restituisce il periodo KijunSen, definito alla creazione dell'indicatore.

SenkouSpanBPeriod

Restituisce il periodo SenkouSpanB.

```
int  SenkouSpanBPeriod() const
```

Valore di ritorno

Restituisce il periodo SenkouSpanB, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,           // simbolo
   ENUM_TIMEFRAMES  period,           // periodo
   int              tenkan_sen,       // periodo di TenkanSen
   int              kijun_sen,        // periodo di KijunSen
   int              senkou_span_b     // periodo di SenkouSpanB
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

tenkan_sen

[in] Periodo di TenkanSen.

kijun_sen

[in] Periodo di KijunSen.

senkou_span_b

[in] Periodo di SenkouSpanB.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

TenkanSen

Restituisce l'elemento di buffer della linea TenkanSen dall'indice specificato.

```
double  TenkanSen(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento buffer della linea TenkanSen .

Valore di ritorno

L'elemento di buffer della linea TenkanSen dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

KijunSen

Restituisce l'elemento di buffer della linea KijunSen dall'indice specificato.

```
double  KijunSen(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento buffer della linea KijunSen .

Valore di ritorno

L'elemento di buffer della linea KijunSen dell'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

SenkouSpanA

Restituisce l'elemento di buffer della linea SenkouSpanA dall'indice specificato.

```
double  SenkouSpanA(
   int  index      // indice
   )
```

Parametri

index

[in] Elemento buffer della linea SenkouSpanA (indice).

Valore di ritorno

L'elemento di buffer della linea SenkouSpanA dell'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

SenkouSpanB

Restituisce l'elemento di buffer della linea SenkouSpanB dall'indice specificato.

```
double  SenkouSpanB(
   int  index      // indice
   )
```

Parametri

index

[in] Elemento buffer della linea SenkouSpanB (indice).

Valore di ritorno

L'elemento di buffer della linea SenkouSpanB dell'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

ChinkouSpan

Restituisce l'elemento di buffer della linea ChinkouSpan dall'indice specificato.

```
double  ChinkouSpan(
   int  index      // indice
   )
```

Parametri

index

[in] Elemento buffer della linea ChinkouSpan (indice).

Valore di ritorno

L'elemento di buffer della linea ChinkouSpan dell'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_ICHIMOKU per Cilchimoku).

CiMA

CIMA è una classe destinata per usare l'indicatore tecnico Moving Average.

Descrizione

La classe CIMA fornisce la creazione, la configurazione e l'accesso ai dati dell'indicatore Moving Average

Dichiarazione

```
class CiMA: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMA

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
MaShift	Restituisce lo slittamento orizzontale
MaMethod	Restituisce il metodo di calcolo della media
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

MaShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  MaShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo della media (valore dell'enumerazione ENUM_MA_METHOD), definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           string,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period,     // periodo medio
   int              ma_shift,      // slittamento
   ENUM_MA_METHOD   ma_method,     // metodo della media
   int              applied        // tipo di prezzo, handle
   )
```

Parametri

string

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ma_shift

[in] Slittamento orizzontale.

ma_method

[in] Metodo di media (ENUM_MA_METHOD, valore dell'enumerazione).

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_MA per Cima).

CiSAR

Cisar è una classe destinata all'utilizzo del indicatore tecnico Parabolic Stop And Reverse System.

Descrizione

Classe Cisar prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Parabolic Stop And Reverse System.

Dichiarazione

```
class CiSAR: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiSAR

I Metodi della Classe per Gruppi

Attributi	
SarStep	Restituisce lo step di incremento dei prezzi
Maximum	Restituisce il valore massimo dello step
Metodo Create	
Create	Crea l'indicatore
Data Access Methods	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

SarStep

Restituisce lo step di incremento dei prezzi.

```
double  SarStep() const
```

Valore di ritorno

Lo step di incremento dei prezzi, definito alla creazione dell'indicatore.

Maximum

Restituisce il valore massimo dello step.

```
double  Maximum() const
```

Valore di ritorno

Il valore massimo dello step, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,      // simbolo
   ENUM_TIMEFRAMES  period,      // periodo
   double           step,        // step
   double           maximum      // coefficiente
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

step

[in] Step per la velocità crescente.

maximum

[in] Coefficienti di inseguimento prezzo.

Valore di ritorno

true - successo, false - non può cambiare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_SAR per Cisar).

CiStdDev

CiStdDev è una classe destinata per utilizzo dell'indicatore tecnico Standard Deviation.

Descrizione

La Classe CiStdDev prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Standard Deviation (deviazione standard).

Dichiarazione

```
class CiStdDev: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStdDev

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
MaShift	Restituisce lo slittamento orizzontale
MaMethod	Restituisce il metodo di calcolo della media
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

MaShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  MaShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo della media (valore dell'enumerazione ENUM_MA_METHOD), definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,         // simbolo
   ENUM_TIMEFRAMES  period,         // periodo
   int              ma_period,      // periodo medio
   int              ma_shift,       // slittamento
   ENUM_MA_METHOD   ma_method,      // metodo della media
   int              applied         // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ma_shift

[in] Slittamento orizzontale.

ma_method

[in] Metodo di media (ENUM_MA_METHOD, valore dell'enumerazione).

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato in caso di successo, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_STDDEV per CiStdDev).

CiDEMA

CiDEMA è una classe destinata all'utilizzo dell' indicatore tecnico Double Exponential Moving Average.

Descrizione

La Classe CiDEMA prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Double Exponential Moving Average.

Dichiarazione

```
class CiDEMA: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDEMA

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
IndShift	Restituisce lo slittamento orizzontale
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

IndShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  IndShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           string,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period,     // periodo medio
   int              ind_shift,     // slittamento
   int              applied        // tipo di prezzo, handle
   )
```

Parametri

string

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ind_shift

[in] Slittamento orizzontale.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento del buffer dell'indice specificato, oppure EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d'indicatore (IND_DEMA per CiDEMA).

CiTEMA

CITEMA è una classe destinata per l'uso dell' indicatore tecnico Triple Exponential Moving Average.

Descrizione

LA Classe CITEMA prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Triple Exponential Moving Average.

Dichiarazione

```
class CiTEMA: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTEMA

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
IndShift	Restituisce lo slittamento orizzontale
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

IndShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  IndShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,         // simbolo
   ENUM_TIMEFRAMES  period,         // periodo
   int              ma_period,      // periodo medio
   int              ma_shift,       // slittamento
   int              applied         // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ma_shift

[in] Slittamento orizzontale.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento del buffer dell'indice specificato, oppure EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_TEMA per CITEMA).

CiFrAMA

CiFrAMA è una classe destinata per l'utilizzo dell' indicatore tecnico Adaptive Frattale Moving Average.

Descrizione

Classe CiFrAMA prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Fractal Adaptive Moving Average.

Dichiarazione

```
class CiFrAMA: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFrAMA

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
IndShift	Restituisce lo slittamento orizzontale
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

IndShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  IndShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,         // simbolo
   ENUM_TIMEFRAMES  period,         // periodo
   int              ma_period,      // periodo medio
   int              ma_shift,       // slittamento
   int              applied         // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ma_shift

[in] Slittamento orizzontale.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dell'indice specificato in caso di successo, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_FRAMA per CiFrAMA).

CiAMA

CiAMA è una classe destinata per l'utilizzo dell' indicatore tecnico Adaptive Moving Average.

Descrizione

La Classe CiAMA prevede la creazione, la configurazione e l'accesso ai dati dell' indicatore Adaptive Moving Average.

Dichiarazione

```
class CiAMA: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAMA

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
FastEmaPeriod	Restituisce il periodo medio per il (veloce)fast EMA
SlowEmaPeriod	Restituisce il periodo medio per il (lento)slow EMA
IndShift	Restituisce lo slittamento orizzontale
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce l'elemento del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

FastEmaPeriod

Restituisce il periodo medio per l'EMA veloce.

```
int  FastEmaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA veloce, definito alla creazione dell'indicatore.

SlowEmaPeriod

Restituisce il periodo medio l' EMA lento.

```
int  SlowEmaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA lento, definito alla creazione dell'indicatore.

IndShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  IndShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           string,              // simbolo
   ENUM_TIMEFRAMES  period,              // periodo
   int              ma_period,           // periodo medio
   int              fast_ema_period,     // periodo fast EMA
   int              slow_ema_period,     // periodo slow EMA
   int              ind_shift,           // slittamento
   int              applied              // tipo di prezzo, handle
   )
```

Parametri

string

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

fast_ema_period

[in] Periodo medio fast EMA.

slow_ema_period

[in] Periodo medio slow EMA.

ind_shift

[in] Slittamento orizzontale.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento del buffer dell'indice specificato, oppure EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d' Indicatore(IND_AMA per CiAMA).

CiVIDyA

CiVIDyA è una classe destinata per usare l' indicatore tecnico Variable Index Dynamic Average.

Descrizione

Classe CiVIDyA prevede la creazione, la configurazione e l'accesso ai dati dell' indicatore Variable Index Dynamic Average.

Dichiarazione

```
class CiVIDyA: public CIndicator
```

Titolo

```
#include <Indicators\Trend.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVIDyA

I Metodi della Classe per Gruppi

Attributi	
CmoPeriod	Restituisce il periodo per il Momentum
EmaPeriod	Restituisce il periodo di averaging
IndShift	Restituisce lo slittamento orizzontale
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Metodo Create	
Create	Crea l'indicatore
Data Access Methods	
Main	Restituisce l'elemento del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

CmoPeriod

Chiude il chart legato alla istanza della classe.

```
int  CmoPeriod() const
```

EmaPeriod

Restituisce il simbolo del chart.

```
int  EmaPeriod() const
```

Valore di ritorno

Restituisce il simbolo del chart legato all'istanza della classe. "" - nessun chart legato.

IndShift

Restituisce lo slittamento orizzontale dell'indicatore.

```
int  IndShift() const
```

Valore di ritorno

Restituisce il valore di slittamento orizzontale, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,          // simbolo
   ENUM_TIMEFRAMES  period,          // periodo
   int              cmo_period,      // periodo momentum
   int              ema_period,      // periodo averaging(medio)
   int              ind_shift,       // slittamento(shift)
   int              applied          // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

cmo_period

[in] Periodo momentum.

ema_period

[in] Periodo medio.

ind_shift

[in] Slittamento orizzontale.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_VIDYA per CiVIDyA).

Classi Oscillator

Questo gruppo di capitoli contiene i dettagli tecnici delle classi oscillatori, così come le descrizioni dei componenti appropriati della libreria standard MQL5.

Classe/gruppo	Descrizione
CiATR	Average True Range
CiBearsPower	Bears Power
CiBullsPower	Bulls Power
CiCCI	Commodity Channel Index
CiChaikin	Chaikin Oscillator
CiDeMarker	DeMarker
CiForce	Force Index
CiMACD	Moving Averages Convergence-Divergence
CiMomentum	Momentum
CiOsMA	Moving Average of Oscillator (MACD histogram)
CiRSI	Relative Strength Index
CiRVI	Relative Vigor Index
CiStochastic	Stochastic Oscillator
CiWPR	Williams' Percent Range
CiTriX	Triple Exponential Moving Averages Oscillator

CiATR

CiATR è una classe destinata per utilizzo dell'indicatore tecnico Average True Range .

Descrizione

La Classe CiATR prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Average True Range.

Dichiarazione

```
class CiATR: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiATR

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_ATR per CiATR).

CiBearsPower

CiBearsPower è una classe destinata per utilizzo dell'indicatore tecnico Bears Power.

Descrizione

Classe CiBearsPower prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Bears Power.

Dichiarazione

```
class CiBearsPower: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBearsPower

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_BEARS per CiBearsPower).

CiBullsPower

CiBullsPower è una classe destinata per utilizzo dell'indicatore tecnico Bulls Power.

Descrizione

La Classe CiBullsPower prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Bulls Power.

Dichiarazione

```
class CiBullsPower: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBullsPower

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce l'elemento del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_BULLS per CiBullsPower).

CiCCI

CiCCI è una classe destinata all'utilizzo dell'indicatore tecnico Commodity Channel Index.

Descrizione

La Classe CiCCI prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Commodity Channel Index.

Dichiarazione

```
class CiCCI: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiCCI

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period,     // periodo medio
   int              applied        // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_CCI per CiCCI).

CiChaikin

CiChaikin è una classe destinata all'utilizzo dell'indicatore tecnico Chaikin Oscillator.

Descrizione

La Classe CiChaikin prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Chaikin Oscillator.

Dichiarazione

```
class CiChaikin: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiChaikin

I Metodi della Classe per Gruppi

Attributi	
FastMaPeriod	Restituisce il periodo medio per il fast MA
SlowMaPeriod	Restituisce il periodo medio per lo slow MA
MaMethod	Restituisce il metodo di calcolo della media
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

FastMaPeriod

Restituisce il periodo medio per l'EMA veloce.

```
int  FastMaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA veloce, definito alla creazione dell'indicatore.

SlowMaPeriod

Restituisce il periodo medio l' EMA lento.

```
int  SlowMaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA lento, definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo di calcolo della media, definita alla creazione dell'indicatore (valore dell' enumerazione ENUM_MA_METHOD).

Applied

Restituisce l'oggetto (tipo di volume) da applicare.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valore di ritorno

Object (tipo di volume) da applicare, definito alla creazione dell'indicatore (valore dell' enumerazioneENUM_APPLIED_VOLUME).

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string               symbol,              // simbolo
   ENUM_TIMEFRAMES      period,              // periodo
   int                  fast_ma_period,      // periodo fast EMA
   int                  slow_ma_period,      // periodo slow EMA
   ENUM_MA_METHOD       ma_method,           // metodo di media
   ENUM_APPLIED_VOLUME  applied              // tipo di volume
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

fast_ma_period

[in] Periodo per fast EMA.

slow_ma_period

[in] Periodo per slow EMA.

ma_method

[in] Metodo di media (ENUM_MA_METHOD, valore dell'enumerazione).

applied

[in] Oggetto (tipo di volume) da applicare (valore dell' enumerazione ENUM_APPLIED_VOLUME).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_CHAIKIN per CiChaikin).

CiDeMarker

CiDeMarker è una classe prevista per l'utilizzo dell'indicatore tecnico DeMarker.

Descrizione

La Classe CiDeMarker prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore DeMarker.

Dichiarazione

```
class CiDeMarker: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDeMarker

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_DEMARKER per CiDeMarker).

CiForce

CiForce è una classe destinata all'utilizzo dell'indicatore tecnico Force Index.

Descrizione

La Classe CiForce prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Force Index.

Dichiarazione

```
class CiForce: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiForce

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
MaMethod	Restituisce il metodo di calcolo della media
Applied	Restituisce l'oggetto (tipo di volume) da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo di calcolo della media, definito alla creazione dell'indicatore.

Applied

Restituisce l'oggetto (tipo di volume) da applicare.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valore di ritorno

Object (tipo di volume) da applicare, definito alla creazione dell'indicatore (valore dell' enumerazioneENUM_APPLIED_VOLUME).

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string                symbol,         // simbolo
   ENUM_TIMEFRAMES       period,         // periodo
   int                   ma_period,      // periodo medio
   ENUM_MA_METHOD        ma_method,      // medoto di media
   ENUM_APPLIED_VOLUME   applied         // tppo di volume
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

ma_method

[in] Metodo di media (ENUM_MA_METHOD, valore dell'enumerazione).

applied

[in] Oggetto (tipo di volume) da applicare (valore dell' enumerazione ENUM_APPLIED_VOLUME).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_FORCE per CiForce).

CiMACD

CiMACD è una classe destinata all'utilizzo dell'indicatore tecnico Moving Averages Convergence-Divergence.

Descrizione

La Classe CiMACD prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Moving Averages Convergence-Divergence.

Dichiarazione

```
class CiMACD: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMACD

I Metodi della Classe per Gruppi

Attributi	
FastEmaPeriod	Restituisce il periodo medio del fast EMA
SlowEmaPeriod	Restituisce il periodo medio per lo slow EMA
SignalPeriod	Restituisce il periodo medio della linea di segnale
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati buffer della linea principale
Signal	Restituisce i dati del buffer della linea signal
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

FastEmaPeriod

Restituisce il periodo medio per l'EMA veloce.

```
int  FastEmaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA veloce, definito alla creazione dell'indicatore.

SlowEmaPeriod

Restituisce il periodo medio l' EMA lento.

```
int  SlowEmaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA lento, definito alla creazione dell'indicatore.

SignalPeriod

Restituisce il periodo medio per la linea di segnale.

```
int  SignalPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea di segnale, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,              // simbolo
   ENUM_TIMEFRAMES  period,              // periodo
   int              fast_ema_period,     // periodo fast EMA
   int              slow_ema_period,     // periodo slow EMA
   int              signal_period,       // periodo di signal
   int              applied              // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

fast_ema_period

[in] Periodo medio fast EMA.

slow_ema_period

[in] Periodo medio slow EMA.

signal_period

[in] Periodo di media della linea Signal.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer della linea principale per l'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento della linea principale del buffer per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Signal

Restituisce l'elemento di buffer della linea di segnale dall'indice specificato.

```
double  Signal(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

L'elemento di buffer della linea di segnale per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_MACD per CiMACD).

CiMomentum

CiMomentum è una classe destinata usando l'indicatore tecnico Momentum.

Descrizione

Classe CiMomentum prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Momentum.

Dichiarazione

```
class CiMomentum: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMomentum

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Applied	Restituisce l'oggetto (tipo di volume) da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce l'elemento del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,         // simbolo
   ENUM_TIMEFRAMES  period,         // periodo
   int              ma_period,      // periodo medio
   int              applied         // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_MOMENTUM per CiMomentum).

CiOsMA

CiOsMA è una classe destinata per l'indicatore tecnico Moving Average of Oscillator (MACD histogram).

Descrizione

La Classe CiOsMA prevede la creazione, la configurazione e l'accesso ai dati della media mobile dell' indicatore tecnico Moving Average of Oscillator (MACD histogram).

Dichiarazione

```
class CiOsMA: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOsMA

I Metodi della Classe per Gruppi

Attributi	
FastEmaPeriod	Restituisce il periodo medio del fast EMA
SlowEmaPeriod	Restituisce il periodo medio per lo slow EMA
SignalPeriod	Restituisce il periodo medio della linea di segnale
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce l'elemento del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

FastEmaPeriod

Restituisce il periodo medio per l'EMA veloce.

```
int  FastEmaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA veloce, definito alla creazione dell'indicatore.

SlowEmaPeriod

Restituisce il periodo medio l' EMA lento.

```
int  SlowEmaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per l' EMA lento, definito alla creazione dell'indicatore.

SignalPeriod

Restituisce il periodo medio per la linea di segnale.

```
int  SignalPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea di segnale, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,              // simbolo
   ENUM_TIMEFRAMES  period,              // periodo
   int              fast_ema_period,     // periodo fast EMA
   int              slow_ema_period,     // periodo slow EMA
   int              signal_period,       // periodo della linea signal
   int              applied              // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

fast_ema_period

[in] Periodo medio fast EMA.

slow_ema_period

[in] Periodo medio slow EMA.

signal_period

[in] Periodo di media della linea Signal.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_OSMA per CiOsMA).

CiRSI

CIRSI è una classe destinata per usare l'indicatore tecnico Relative Strength Index.

Descrizione

La Classe CIRSI prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Relative Strength Index.

Dichiarazione

```
class CiRSI: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRSI

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce l'elemento del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period,     // periodo medio
   int              applied        // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_RSI per CIRSI).

CiRVI

CiRVI è una classe destinata all' utilizzo dell'indicatore tecnico Relative Vigor Index.

Descrizione

La Classe CiRVI prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Relative Vigor Index.

Dichiarazione

```
class CiRVI: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRVI

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Metodo Create	
Create	Crea l'indicatore
Data Access Methods	
Main	Restituisce i dati buffer della linea principale
Signal	Restituisce i dati del buffer della linea signal
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,        // simbolo
   ENUM_TIMEFRAMES  period,        // periodo
   int              ma_period      // periodo medio
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer della linea principale per l'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell'elemento buffer.

Valore di ritorno

Elemento della linea principale del buffer per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Signal

Restituisce l'elemento di buffer della linea di segnale dall'indice specificato.

```
double  Signal(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

L'elemento di buffer della linea di segnale per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_RVI per CiRVI).

CiStochastic

CiStochastic è una classe destinata all'utilizzo dell'indicatore tecnico Stochastic Oscillator.

Descrizione

La Classe CiStochastic prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Stochastic Oscillator.

Dichiarazione

```
class CiStochastic: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStochastic

I Metodi della Classe per Gruppi

Attributi	
Kperiod	Restituisce il periodo medio per la linea %K
Dperiod	Restituisce il periodo medio per la linea %D
Slowing	Restituisce il periodo di slowing
MaMethod	Restituisce il metodo di calcolo della media
PriceField	Tipo di prezzo da applicare (Low/High o Close/Close)
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati buffer della linea principale
Signal	Restituisce i dati del buffer della linea signal
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Kperiod

Restituisce il periodo medio per la linea %K.

```
int  Kperiod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea %K, definito alla creazione dell'indicatore.

Dperiod

Restituisce il periodo medio per la linea %D.

```
int  Dperiod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea %D, definita alla creazione dell'indicatore.

Slowing

Restituisce il periodo di rallentamento.

```
int  Slowing() const
```

Valore di ritorno

Restituisce il periodo di slowing, definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo di calcolo della media, definita alla creazione dell'indicatore (valore dell'enumerazione ENUM_MA_METHOD).

PriceField

Restituisce l'oggetto a cui applicare (Low/High o Close/Close).

```
ENUM_STO_PRICE  PriceField() const
```

Valore di ritorno

L'oggetto (Low/High o Close/Close) da applicare, definito alla creazione dell'indicatore (valore dell'enumerazione ENUM_STO_PRICE).

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,          // simbolo
   ENUM_TIMEFRAMES  period,          // periodo
   int              Kperiod,         // periodo %K
   int              Dperiod,         // periodo %D
   int              slowing,         // periodo di slowing
   ENUM_MA_METHOD   ma_method,       // periodo di media
   ENUM_STO_PRICE   price_field      // applicazione
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

Kperiod

[in] Periodo medio di %K dell'indicatore.

Dperiod

[in] Periodo medio di %D dell'indicatore.

slowing

[in] Periodo di slowing.

ma_method

[in] Metodo di media (ENUM_MA_METHOD, valore dell'enumerazione).

price_field

[in] Oggetto (Low/High o Close/Close) da applicare (valore dell'enumerazione ENUM_STO_PRICE).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer della linea principale per l'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento della linea principale del buffer per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Signal

Restituisce l'elemento di buffer della linea di segnale dall'indice specificato.

```
double  Signal(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

L'elemento di buffer della linea di segnale per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_STOCHASTIC per CiStochastic).

CiTriX

Citrix è una classe destinata per usarel'indicatore tecnico Triple Exponential Moving Averages Oscillator.

Descrizione

La classe di Citrix prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Triple Exponential Moving Averages Oscillator.

Dichiarazione

```
class CiTriX: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTriX

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,         // simbolo
   ENUM_TIMEFRAMES  period,         // periodo
   int              ma_period,      // periodo medio
   int              applied         // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

applied

[in] Tipo prezzo dell' handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_TRIX per Citrix).

CiWPR

CiWPR è una classe destinata all'utilizzo dell' indicatore tecnico Williams' Percent Range.

Descrizione

La Classe CiWPR prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Williams' Percent Range.

Dichiarazione

```
class CiWPR: public CIndicator
```

Titolo

```
#include <Indicators\Oscilators.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiWPR

I Metodi della Classe per Gruppi

Attributi	
CalcPeriod	Restituisce il periodo di calcolo
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

CalcPeriod

Restituisce il periodo per il calcolo.

```
int  CalcPeriod() const
```

Valore di ritorno

Restituisce il periodo per il calcolo, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,          // simbolo
   ENUM_TIMEFRAMES  period,          // periodo
   int              calc_period      // periodo di calcolo
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

calc_period

[in] Periodo di calcolo.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_WPR per CiWPR).

Volume Indicators

Questo gruppo di capitoli contiene i dettagli tecnici delle classi indicatore del Volume e le descrizioni di tutti i componenti chiave appropriati della Libreria Standard MQL5.

Classe/gruppo	Descrizione
CiAD	Accumulation/Distribution
CiMFI	Money Flow Index
CiOBV	On Balance Volume
CiVolumes	Volumes

CiAD

CiAD è una classe destinata all'utilizzo dell'indicatore tecnico Accumulation/Distribution.

Descrizione

La classe Ciad prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Accumulation/Distribution.

Dichiarazione

```
class CiAD: public CIndicator
```

Titolo

```
#include <Indicators\Volumes.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAD

I Metodi della Classe per Gruppi

Attributi	
Applied	Restituisce il periodo di calcolo
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Restituisce il tipo di volume da applicare.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valore di ritorno

Tipo di volume da applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string               symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES      period,      // periodo(period)
   ENUM_APPLIED_VOLUME  applied      // tipo di volume
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

applied

[in] Il tipo di volume da applicare (valore dell' enumerazione ENUM_APPLIED_VOLUME).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_AD per Ciad).

CiMFI

CiMFI è una classe destinata per utilizzo dell'indicatore tecnico Money Flow Index.

Descrizione

La Classe CiMFI prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Money Flow Index.

Dichiarazione

```
class CiMFI: public CIndicator
```

Titolo

```
#include <Indicators\Volumes.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMFI

I Metodi della Classe per Gruppi

Attributi	
MaPeriod	Restituisce il periodo di averaging
Applied	Restituisce il tipo di volume da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Restituisce il periodo di media.

```
int  MaPeriod() const
```

Valore di ritorno

Restituisce il periodo medio(averaging), definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di volume da applicare.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valore di ritorno

Tipo di volume da applicare, definito alla creazione dell'indicatore (valore di enumerazione ENUM_APPLIED_VOLUME).

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string                symbol,         // simbolo
   ENUM_TIMEFRAMES       period,         // periodo
   int                   ma_period,      // periodo medio
   ENUM_APPLIED_VOLUME   applied         // tppo di volume
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

ma_period

[in] Periodo medio.

applied

[in] Il tipo di volume da applicare (valore dell' enumerazione ENUM_APPLIED_VOLUME).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_MFI per CiMFI).

CiOBV

CiOBV è una classe destinata per utilizzo dell'indicatore tecnico On Balance Volume.

Descrizione

LA Classe CiOBV prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore On Balance Volume.

Dichiarazione

```
class CiOBV: public CIndicator
```

Titolo

```
#include <Indicators\Volumes.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOBV

I Metodi della Classe per Gruppi

Attributi	
Applied	Restituisce il tipo di volume da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Restituisce il tipo di volume da applicare.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valore di ritorno

Tipo di volume da applicare, definito alla creazione dell'indicatore (valore di enumerazione ENUM_APPLIED_VOLUME).

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string               symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES      period,      // periodo(period)
   ENUM_APPLIED_VOLUME  applied      // tipo di volume
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

applied

[in] Il tipo di volume da applicare (valore dell' enumerazione ENUM_APPLIED_VOLUME).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_OBV per CiOBV).

CiVolumes

CiVolumes è una classe destinata per l'utilizzo dell'indicatore tecnico Volumes.

Descrizione

La Classe CiVolumes prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Volumi(volumes).

Dichiarazione

```
class CiVolumes: public CIndicator
```

Titolo

```
#include <Indicators\Volumes.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVolumes

I Metodi della Classe per Gruppi

Attributi	
Applied	Restituisce il tipo di volume da applicare
Metodo Create	
Create	Crea l'indicatore
Data Access Methods	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Restituisce il tipo di volume da applicare.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valore di ritorno

Tipo di volume da applicare, definito alla creazione dell'indicatore (valore di enumerazione ENUM_APPLIED_VOLUME).

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string               symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES      period,      // periodo(period)
   ENUM_APPLIED_VOLUME  applied      // tipo di volume
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

applied

[in] Il tipo di volume da applicare (valore dell' enumerazione ENUM_APPLIED_VOLUME).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_VOLUMES per CiVolumes).

Bill Williams Indicators

Questo gruppo di capitoli contiene i dettagli tecnici delle classi indicatore Bill Williams e le descrizioni di tutti i componenti appropriati della libreria standard MQL5.

Classe/gruppo	Descrizione
CiAC	Accelerator Oscillator
CiAlligator	Alligator
CiAO	Awesome Oscillator
CiFractals	Fractals
CiGator	Gator Oscillator
CiBWMFI	Market Facilitation Index

CiAC

CiAC è una classe destinata all'utilizzo dell'indicatore tecnico Accelerator Oscillator.

Descrizione

La Classe CiAC fornisce la creazione, la configurazione e l'accesso ai dati dell'indicatore Accelerator Oscillator.

Dichiarazione

```
class CiAC: public CIndicator
```

Titolo

```
#include <Indicators\BillWilliams.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAC

I Metodi della Classe per Gruppi

Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo d'indicatore (IND_AC per CiAC).

CiAlligator

CiAlligator è una classe destinata all' utilizzo dell'indicatore tecnico Alligator.

Descrizione

La Classe CiAlligator prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Alligator.

Dichiarazione

```
class CiAlligator: public CIndicator
```

Titolo

```
#include <Indicators\BillWilliams.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAlligator

I Metodi della Classe per Gruppi

Attributi	
JawPeriod	Restituisce il periodo medio per la linea di Jaws
JawShift	Restituisce lo slittamento orizzontale della linea di Jaws
TeethPeriod	Restituisce il periodo medio per la linea Teeth
TeethShift	Restituisce lo slittamento orizzontale della linea Teeth
LipsPeriod	Restituisce il periodo medio per la linea Lips
LipsShift	Restituisce lo slittamento orizzontale della linea di Lips
MaMethod	Restituisce il metodo di calcolo della media
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Jaw	Restituisce i dati di buffer del buffer della linea Jaws
Teeth	Restituisce i dati di buffer del buffer della linea Teeth

Attributi	
Lips	Restituisce i dati di buffer del buffer della linea Lips
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

JawPeriod

Restituisce il periodo medio per la linea Jaw.

```
int  JawPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea Jaw, definito alla creazione dell'indicatore.

JawShift

Restituisce lo slittamento orizzontale della linea Jaws.

```
int  JawShift() const
```

Valore di ritorno

Slittamento orizzontale della linea di Jaws, definito alla creazione dell'indicatore.

TeethPeriod

Restituisce il periodo medio per la linea Teeth.

```
int  TeethPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea Teeth, definito alla creazione dell'indicatore.

TeethShift

Restituisce lo slittamento orizzontale della linea Teeth.

```
int  TeethShift() const
```

Valore di ritorno

Slittamento orizzontale della linea Teeth, definito alla creazione dell'indicatore.

LipsPeriod

Restituisce il periodo medio per la linea Lips.

```
int  LipsPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea Lips, definito alla creazione dell'indicatore.

LipsShift

Restituisce lo slittmento orizzontale della linea Lips.

```
int  LipsShift() const
```

Valore di ritorno

Slittamento orizzontale della linea Lips, definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo di calcolo della media, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,           // simbolo
   ENUM_TIMEFRAMES  period,           // periodo
   int              jaw_period,       // periodo jaws
   int              jaw_shift,        // slittamento jaws
   int              teeth_period,     // periodo teeth
   int              teeth_shift,      // slittamento teeth
   int              lips_period,      // periodo lips
   int              lips_shift,       // slittamento lips
   ENUM_MA_METHOD   ma_method,        // metodo di averaging(media)
   int              applied           // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

jaw_period

[in] periodo di averaging di Jaws.

jaw_shift

[in] slittamento orizzontale Jaws.

teeth_period

[in] periodo di averaging di Teeth.

teeth_shift

[in] slittamento orizzontale Teeth.

lips_period

[in] periodo di averaging di Lips.

lips_shift

[in] slittamento orizzontale Lips.

ma_method

[in] Metodo di media mobile (ENUM_MA_METHOD, valore dell'enumerazione).

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Jaw

Restituisce l'elemento di buffer della linea Jaws dall'indice specificato.

```
double  Jaw(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell'elemento Linea di Jaws .

Valore di ritorno

L'elemento di buffer della linea Jaws dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Teeth

Restituisce l'elemento di buffer della linea Teeth dall'indice specificato.

```
double  Teeth(
   int  index      // indice
   )
```

Parametri

index

[In] Indice elemento buffer della linea Teeth.

Valore di ritorno

L'elemento buffer della linea Teeth dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Lips

Restituisce l'elemento di buffer della linea Lips dall'indice specificato.

```
double  Lips(
   int  index      // indice
   )
```

Parametri

index

[in] Linea Lips, indice elemento buffer.

Valore di ritorno

L'elemento di buffer della linea Lips dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_ALLIGATOR per CiAlligator).

CiAO

CiAO è una classe destinata all'utilizzo dell'indicatore tecnico Awesome Oscillator.

Descrizione

La Classe CiAO prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Awesome Oscillator.

Dichiarazione

```
class CiAO: public CIndicator
```

Titolo

```
#include <Indicators\BillWilliams.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAO

I Metodi della Classe per Gruppi

Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce l'elemento del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento del buffer dell'indice specificato, oppure EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_AO per Ciao).

CiFractals

CiFractals è una classe destinata all'utilizzo dell'indicatore tecnico Fractals.

Descrizione

La Classe CiFractals prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Fractals.

Dichiarazione

```
class CiFractals: public CIndicator
```

Titolo

```
#include <Indicators\BillWilliams.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFractals

I Metodi della Classe per Gruppi

Create	
Create	Crea l'indicatore
Accesso ai dati	
Upper	Restituisce i dati del buffer superiore
Lower	Restituisce i dati del buffer inferiore
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES  period       // periodo(period)
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Upper

Restituisce l'elemento del buffer superiore per indice specificato.

```
double  Upper(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento del buffer superiore.

Valore di ritorno

L'elemento del buffer superiore dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Lower

Restituisce l'elemento del buffer inferiore per indice specificato.

```
double  Lower(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento del buffer inferiore.

Valore di ritorno

L'elemento del buffer inferiore dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_FRACTALS per CiFractals).

CiGator

CiGator è una classe destinata per utilizzo dell'indicatore tecnico Gator Oscillator.

Descrizione

La Classe CiGator prevede la creazione, la configurazione e l'accesso ai dati dell'indicatore Gator Oscillator.

Dichiarazione

```
class CiGator: public CIndicator
```

Titolo

```
#include <Indicators\BillWilliams.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiGator

I Metodi della Classe per Gruppi

Attributi	
JawPeriod	Restituisce il periodo medio per la linea di Jaws
JawShift	Restituisce lo slittamento orizzontale della linea di Jaws
TeethPeriod	Restituisce il periodo medio per la linea Teeth
TeethShift	Restituisce lo slittamento orizzontale della linea Teeth
LipsPeriod	Restituisce il periodo medio per la linea Lips
LipsShift	Restituisce lo slittamento orizzontale della linea di Lips
MaMethod	Restituisce il metodo di calcolo della media
Applied	Restituisce il tipo di prezzo o l' handle da applicare
Metodo Create	
Create	Crea l'indicatore
Data Access Methods	
Upper	Restituisce i dati del buffer superiore

Attributi	
Lower	Restituisce i dati del buffer inferiore
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

JawPeriod

Restituisce il periodo medio per la linea di Jaws.

```
int  JawPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea di Jaws, definito alla creazione dell'indicatore.

JawShift

Restituisce lo slittamento orizzontale della linea Jaws.

```
int  JawShift() const
```

Valore di ritorno

Slittamento orizzontale della linea di Jaws, definito alla creazione dell'indicatore.

TeethPeriod

Restituisce il periodo medio per la linea Teeth.

```
int  TeethPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea Teeth, definito alla creazione dell'indicatore.

TeethShift

Restituisce lo slittamento orizzontale della linea Teeth.

```
int  TeethShift() const
```

Valore di ritorno

Slittamento orizzontale della linea Teeth, definito alla creazione dell'indicatore.

LipsPeriod

Restituisce il periodo medio per la linea Lips.

```
int  LipsPeriod() const
```

Valore di ritorno

Restituisce il periodo medio per la linea Lips, definito alla creazione dell'indicatore.

LipsShift

Restituisce lo slittmento orizzontale della linea Lips.

```
int  LipsShift() const
```

Valore di ritorno

Slittamento orizzontale della linea Lips, definito alla creazione dell'indicatore.

MaMethod

Restituisce il metodo di averaging(di media).

```
ENUM_MA_METHOD  MaMethod() const
```

Valore di ritorno

Restituisce il metodo di calcolo della media, definito alla creazione dell'indicatore.

Applied

Restituisce il tipo di prezzo o l'handle da applicare.

```
int  Applied() const
```

Valore di ritorno

Tipo prezzo o handleda applicare, definito alla creazione dell'indicatore.

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string           symbol,           // simbolo
   ENUM_TIMEFRAMES  period,           // periodo
   int              jaw_period,       // periodo jaws
   int              jaw_shift,        // slittamento jaws
   int              teeth_period,     // periodo teeth
   int              teeth_shift,      // slittamento teeth
   int              lips_period,      // periodo lips
   int              lips_shift,       // slittamento lips
   ENUM_MA_METHOD   ma_method,        // metodo di averaging(media)
   int              applied           // tipo di prezzo, handle
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

jaw_period

[in] periodo di averaging di Jaws.

jaw_shift

[in] slittamento orizzontale Jaws.

teeth_period

[in] periodo di averaging di Teeth.

teeth_shift

[in] slittamento orizzontale Teeth.

lips_period

[in] periodo di averaging di Lips.

lips_shift

[in] slittamento orizzontale Lips.

ma_method

[in] Metodo di media (ENUM_MA_METHOD, valore dell'enumerazione).

applied

[in] Tipo di prezzo o handle da applicare.

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Upper

Restituisce l'elemento del buffer superiore per indice specificato.

```
double  Upper(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento del buffer superiore.

Valore di ritorno

L'elemento del buffer superiore per l'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Lower

Restituisce l'elemento del buffer inferiore per indice specificato.

```
double  Upper(
   int  index      // indice
   )
```

Parametri

index

[in] Indice elemento del buffer inferiore.

Valore di ritorno

L'elemento del buffer inferiore dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_GATOR per CiGator).

CiBWMFI

CiBWMFI è una classe destinata all'utilizzo del Market Facilitation Index per indicatore tecnico Bill Williams.

Descrizione

La Classe CiBWMFI prevede la creazione, la configurazione e l'accesso ai dati del Market Facilitation Index per indicatore Bill Williams.

Dichiarazione

```
class CiBWMFI: public CIndicator
```

Titolo

```
#include <Indicators\BillWilliams.mqh>
```

Gerarchia di ereditarietà

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBWMFI

I Metodi della Classe per Gruppi

Attributi	
Applied	Restituisce il tipo di volume da applicare
Create	
Create	Crea l'indicatore
Accesso ai dati	
Main	Restituisce i dati del buffer
Input/output	
virtual Type	Metodo di identificazione virtuale

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

Metodi ereditati dalla classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Metodi ereditati dalla classe CArrayObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Metodi ereditati dalla classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Metodi ereditati dalla classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Restituisce il tipo di volume da applicare.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valore di ritorno

Tipo di volume da applicare, definito alla creazione dell'indicatore (valore di enumerazione ENUM_APPLIED_VOLUME).

Create

Crea l'indicatore con i parametri specificati. Usa Refresh() e GetData() per aggiornare ed ottenere i valori dell'indicatore.

```
bool  Create(
   string                symbol,      // simbolo(symbol)
   ENUM_TIMEFRAMES       period,      // periodo(period)
   ENUM_APPLIED_VOLUME   applied      // tipo di volume
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES valore enumerazione).

applied

[in] Tipo di volume da applicare (valore di enumerazione ENUM_APPLIED_VOLUME).

Valore di ritorno

true - successo, false - non si può creare l'indicatore.

Main

Restituisce l'elemento buffer dell'indice specificato.

```
double  Main(
   int  index      // indice
   )
```

Parametri

index

[in] Indice dell' elemento del Buffer.

Valore di ritorno

Elemento Buffer dall'indice specificato, o EMPTY_VALUE se non ci sono dati corretti.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_BWMFI per CiBWMFI).

CiCustom

CiCustom è una classe destinata per l'utilizzo degli indicatori tecnici personalizzati.

Descrizione

La Classe CiCustom prevede la creazione, la configurazione e l'accesso ai dati di un indicatore personalizzato.

Dichiarazione

```
class CiCustom: public CIndicator
```

Titolo

```
#include <Indicators\Custom.mqh>
```

I Metodi della Classe per Gruppi

Attributi	
NumBuffers	Imposta il numero di buffer
NumParams	Ottiene il numero di parametri utilizzati durante la creazione di un indicatore
ParamType	Ottiene il tipo del parametro specificato
ParamLong	Ottiene il valore del parametro specificato di tipo integer
ParamDouble	Ottiene il valore del parametro specificato di tipo double
ParamString	Ottiene il valore del parametro specificato di tipo string
Input/output	
virtual Type	Metodo di identificazione virtuale

NumBuffers

Imposta il numero del buffer.

```
bool  NumBuffers(
```

Valore di ritorno

true - successo, false - non può impostare il numero necessario di buffer.

NumParams

Ottiene il numero di parametri utilizzati durante la creazione di un indicatore.

```
int  NumParams() const
```

Valore di ritorno

Numero di parametri, utilizzati nella creazione dell'indicatore.

ParamType

Ottiene il tipo di parametro.

```
ENUM_DATATYPE  ParamType(
   int  index      // indice
   ) const
```

Parametri

index

[in] Indice parametro.

Valore di ritorno

Restituisce il tipo di dati del parametro specificato, usato nella creazione dell'indicatore.

Nota

Se l'indice parametro non è valido, restituisce WRONG_VALUE.

ParamLong

Ottiene il valore del parametro specificato di tipo long.

```
long  ParamLong(
   int  index      // indice
   ) const
```

Parametri

index

[in] Indice parametro.

Valore di ritorno

Il valore del parametro specificato di tipo long, utilizzato nella creazione dell'indicatore.

Nota

Se l'indice del parametro è valido o il parametro non è di tipo long, restituisce 0.

ParamDouble

Ottiene il valore del parametro specificato di tipo double.

```
double  ParamDouble(
   int  index      // indice
   ) const
```

Parametri

index

[in] Indice parametro.

Valore di ritorno

Il valore del parametro specificato di tipo double, usato nella creazione dell'indicatore.

Nota

Se l'indice del parametro è valido o il tipo di parametro non è di tipo double, restituisce EMPTY_VALUE.

ParamString

Ottiene il valore del parametro specificato di tipo string.

```
string  ParamString(
   int  index      // indice
   ) const
```

Parametri

index

[in] Indice parametro.

Valore di ritorno

Il valore del parametro string specificato, utilizzato nella realizzazione dell'indicatore.

Nota

Se il numero non è valido o il parametro non è di tipo string, restituisce una stringa vuota.

Type

Metodo di identificazione virtuale.

```
virtual int  Type() const
```

Valore di ritorno

Tipo di Indicatore (IND_CUSTOM per CiCustom).

Classi di Trading

Questa sezione contiene i dettagli tecnici di lavoro con le classi di trade e la descrizione dei componenti rilevanti della libreria standard MQL5.

L'utilizzo delle classi di trade consente di risparmiare tempo durante la creazione di programmi personalizzati (Expert Advisors).

MQL5 standard Library (in termini di classi di trade) è inserita nella directory di lavoro del terminale, nella cartella Include\Trade.

Classe/Gruppo	Descrizione
CAccountInfo	Classe di lavoro con le proprietà degli account di trade
CSymbolInfo	Classe di lavoro con le proprietà di strumento di trade
COrderInfo	Classe di lavoro con le proprietà di ordini pendenti
CHistoryOrderInfo	Classe di lavoro con le proprietà di ordini history (cronistoria ordini).
CPositionInfo	Classe di lavoro con le proprietà di posizione aperte
CDealInfo	Classe per lavorare con le proprietà di hostory affare (cronistoria deals)
CTrade	Class per l'esecuzione di operazioni di trade
CTerminalInfo	Classe per ottenere le proprietà dell'ambiente del terminal

CAccountInfo

CAccountInfo è una classe per il facile accesso alle proprietà del trade account attualmente aperto.

Descrizione

La Classe CAccountInfo offre un facile accesso alle proprietà dell'account di trade attualmente aperto.

Dichiarazione

```
class CAccountInfo : public CObject
```

Titolo

```
#include <Trade\AccountInfo.mqh>
```

Gerarchia di ereditarietà

CObject

CAccountInfo

I metodi di classe per gruppi

L'accesso alla proprietà di tipo integer	
Login	Ottiene il numero di conto
TradeMode	Ottiene la modalità di trade
TradeModeDescription	Ottiene la modalità di trade come una stringa
Leverage	Ottiene la quantità di leva finanziaria ottenuta
StopoutMode	Ottiene la modalità di impostazione di stop out
StopoutModeDescription	Ottiene il modo di impostazioni stop out come stringa
TradeAllowed	Ottiene la flage di permesso al trading
TradeExpert	Ottiene la flag di permesso al trading automatizzato
LimitOrders	Ottiene il numero massimo di ordini pendenti consentiti
MarginMode	Ottiene la modalità di calcolo del margine
MarginModeDescription	Ottiene la modalità di calcolo del margine come un stringa
L'accesso alla proprietà di tipo double	
Balance	Ottiene il balance de conto
Credit	Ottiene la quantità credito dato
Profit	Ottiene la quantità di profitto corrente sul conto

L'accesso alla proprietà di tipo integer	
Equity	Ottiene la quantità della corrente equità sul conto
Margin	Ottiene l'ammontare del margine riservato
FreeMargin	Ottiene la quantità di margine libero
MarginLevel	Ottiene il livello di margine
MarginCall	Ottiene il livello di margine per il deposito
MarginStopOut	Ottiene il livello di margine di Stop Out
L'accesso alla proprietà testuali	
Name	Ottiene il nome del cliente
Server	Ottiene il nome del trade server
Currency	Ottiene il nome della valuta di deposito
Company	Ottiene il nome della società che serve un account
L'accesso alle funzioni API MQL5	
InfoInteger	Ottiene il valore della proprietà di tipo integer specificata
InfoDouble	Ottiene il valore della proprietà di tipo double specificata
InfoString	Ottiene il valore della proprietà di tipo stringa specificata
Metodi aggiuntivi	
OrderProfitCheck	Ottiene il profitto valutato in base ai parametri passati
MarginCheck	Ottiene la quantità di margine richiesto per eseguire un'operazione di trade
FreeMarginCheck	Ottiene la quantità di margine libero rimasto dopo l'esecuzione dell'operazione di trade
MaxLotCheck	Ottiene il possibile volume massimo dell' attività di trade

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Login

Ottiene il numero di conto.

```
long  Login() const
```

Valore di ritorno

Numero di conto.

TradeMode

Ottiene la modalità di trade.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

Valore di ritorno

Modalità di trade dall'enumerazione ENUM_ACCOUNT_TRADE_MODE.

TradeModeDescription

Ottiene la modalità di trade sotto forma di stringa.

```
string  TradeModeDescription() const
```

Valore di ritorno

Modalità di trade, come stringa.

Leverage

Ottiene la quantità di leva data.

```
long  Leverage() const
```

Valore di ritorno

Quantità di leva data.

StopoutMode

Ottiene la modalità della specificazione livello di stop out.

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

Valore di ritorno

La modalità di impostazione dello stop out dall'enumerazione ENUM_ACCOUNT_STOPOUT_MODE.

StopoutModeDescription

Ottiene la modalità della specificazione livello di stop out come stringa.

```
string  StopoutModeDescription() const
```

Valore di ritorno

Ottiene la modalità di impostazione dello stop out come stringa

MarginMode

Ottiene la modalità di calcolo del margine.

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

Valore di ritorno

La modalità di calcolo del margine dall'enumerazione ENUM_ACCOUNT_MARGIN_MODE.

MarginModeDescription

Ottiene la modalità di calcolo del margine come stringa.

```
string  MarginModeDescription() const
```

Valore di ritorno

Modalità di calcolo del margine come stringa.

TradeAllowed

Ottiene la flag per il permesso al trade.

```
bool  TradeAllowed() const
```

Valore di ritorno

Flag per il permesso al trade.

TradeExpert

Ottiene la flag per il permesso al trading automatizzato.

```
bool  TradeExpert() const
```

Valore di ritorno

Flag per il permesso al trade automatizzato.

LimitOrders

Ottiene il numero massimo di ordini pendenti consentiti

```
int  LimitOrders() const
```

Valore di ritorno

Il numero massimo di ordini pendenti consentiti.

Nota

0 - nessun limite.

Balance

Ottiene il bilancio del conto.

```
double  Balance() const
```

Valore di ritorno

Il saldo(balance) del conto (in valuta di deposito).

Credit

Ottiene la quantità di credito dato.

```
double  Credit() const
```

Valore di ritorno

Importo del credito dato (in valuta di deposito).

Profit

Ottiene la quantità del corrente profitto sul conto.

```
double  Profit() const
```

Valore di ritorno

Quantità di profitto corrente sul conto (in valuta di deposito).

Equity

Ottiene l'importo della corrente equità netta sul conto.

```
double  Equity() const
```

Valore di ritorno

Importo della corrente equità sul conto (in valuta di deposito).

Margin

Ottiene l'ammontare del margine riservato.

```
double  Margin() const
```

Valore di ritorno

Ammontare del margine riservato (in valuta di deposito).

FreeMargin

Ottiene la quantità di margine libero.

```
double  FreeMargin() const
```

Valore di ritorno

Quantità di margine libero (in valuta di deposito).

MarginLevel

Ottiene il livello del margine.

```
double  MarginLevel() const
```

Valore di ritorno

Livello di margine.

MarginCall

Ottiene il livello di margine per un deposito.

```
double  MarginCall() const
```

Valore di ritorno

Livello di margine per un deposito.

MarginStopOut

Ottiene il livello di margine di Stop Out.

```
double  MarginStopOut() const
```

Valore di ritorno

Livello di margine per lo Stop Out.

Name

Ottiene il nome del cliente.

```
string  Name() const
```

Valore di ritorno

Nome del cliente.

Server

Ottiene il nome del trade server.

```
string  Server() const
```

Valore di ritorno

Nome del server di trade.

Currency

Ottiene il nome della valuta di deposito.

```
string  Currency() const
```

Valore di ritorno

Nome della valuta di deposito.

Company

Ottiene il nome della società, che serve un account.

```
string  Company() const
```

Valore di ritorno

Nome azienda che serve l'account.

InfoInteger

Ottiene il valore della specificata proprietà di tipo integer.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // ID proprietà
   ) const
```

Parametri

prop_id

[in] Identificatore della proprietà. Il valore può essere uno dei valori dell'enumerazione ENUM_ACCOUNT_INFO_INTEGER.

Valore di ritorno

Valore di tipo long.

InfoDouble

Ottiene il valore della proprietà di tipo double specificata.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // ID della proprietà
   ) const
```

Parametri

prop_id

[in] Identificatore della proprietà. Il valore può essere uno dei valori dell'enumerazione ENUM_ACCOUNT_INFO_DOUBLE.

Valore di ritorno

Valore di tipo double.

InfoString

Ottiene il valore della proprietà di tipo string, specificata.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // ID proprietà
   ) const
```

Parametri

prop_id

[in] Identificatore della proprietà. Il valore può essere uno dei valori dell'enumerazione ENUM_ACCOUNT_INFO_STRING.

Valore di ritorno

Valore di tipostring.

OrderProfitCheck

La funzione calcola il profitto per il corrente account, in base ai parametri passati. La funzione è utilizzata per la pre-valutazione del risultato di un'operazione di trade. Il valore viene restituito nella valuta del conto.

```
double  OrderProfitCheck(
   const string        symbol,              // simbolo
   ENUM_ORDER_TYPE     trade_operation,     // tipo di ordine (ORDER_TYPE_BUY oppure (
   double              volume,              // volume
   double              price_open,          // prezzo di apertura della posizione
   double              price_close          // prezzo di chiusura della posizione
   ) const
```

Parametri

symbol

[in] Simbolo per l'operazione di trade.

trade_operation

[in] Tipo di operazione di trade dall'enumerazione ENUM_ORDER_TYPE.

volume

[in] Volume dell'operazione di trade.

price_open

[in] Prezzo Open.

price_close

[in] Prezzo Close.

Valore di ritorno

In caso di successo, restituisce quantità di profitto o EMPTY_VALUE in caso di errore.

MarginCheck

Ottiene l'ammontare del margine, necessario per l'operazione di trade.

```
double  MarginCheck(
   const string        symbol,              // simbolo
   ENUM_ORDER_TYPE     trade_operation,     // tipo di ordine (ORDER_TYPE_BUY oppure (
   double              volume,              // volume
   double              price                // prezzo
   ) const
```

Parametri

symbol

[in] Simbolo per l'operazione di trade.

trade_operation

[in] Tipo di operazione di trade dall'enumerazione ENUM_ORDER_TYPE.

volume

[in] Volume dell'operazione di trade.

price

[in] Prezzo dell'operazione di trade.

Valore di ritorno

Importo del margine richiesto per l'operazione di trade.

FreeMarginCheck

Ottiene la quantità di margine libero lasciato dopo l'operazione di trade.

```
double  FreeMarginCheck(
   const string        symbol,              // simbolo
   ENUM_ORDER_TYPE     trade_operation,     // tipo di ordine (ORDER_TYPE_BUY oppure (
   double              volume,              // volume
   double              price                // prezzo
   ) const
```

Parametri

symbol

[in] Simbolo per l'operazione di trade.

trade_operation

[in] Tipo di operazione di trade dall'enumerazione ENUM_ORDER_TYPE.

volume

[in] Volume dell'operazione di trade.

price

[in] Prezzo dell'operazione di trade.

Valore di ritorno

Quantità di margine libero rimasto dopo l'operazione di trade.

MaxLotCheck

Ottiene il possibile volume massimo dell'operazione di trade.

```
double  MaxLotCheck(
   const string        symbol,              // simbolo
   ENUM_ORDER_TYPE     trade_operation,     // tipo di ordine (ORDER_TYPE_BUY oppure (
   double              price,               // prezzo
   double              percent=100          // percentuale di margine disponibile (de
   ) const
```

Parametri

symbol

[in] Simbolo per l'operazione di trade.

trade_operation

[in] Tipo di operazione di trade dall'enumerazione ENUM_ORDER_TYPE.

price

[in] Prezzo dell'operazione di trade.

percent=100

[in] Percentuale di margine disponibile (in%) da utilizzare per l'operazione di trade.

Valore di ritorno

Massimo volume possibile dell' operazione di trade.

CSymbolInfo

CSymbolInfo è una classe per il facile accesso alle proprietà del simbolo.

Descrizione

La Classe CSymbolInfo consente di accedere alle proprietà del simbolo.

Dichiarazione

```
class CSymbolInfo : public CObject
```

Titolo

```
#include <Trade\SymbolInfo.mqh>
```

Gerarchia di ereditarietà

CObject

CSymbolInfo

I metodi di classe per gruppi

Controlling	
Refresh	Aggiorna i dati del simbolo
RefreshRates	Aggiorna le quotazioni dei simboli
Proprietà	
Name	Ottiene/Imposta il nome del simbolo
Select	Ottiene/Imposta la flag "Market Watch" del simbolo
IsSynchronized	Controlla la sincronizzazione del simbolo con il server
Volumes	
Volume	Ottiene il volume del precedente deal
VolumeHigh	Ottiene il massimo volume per il giorno
VolumeLow	Ottiene il volume minimo per il giorno
Miscellaneous	
Time	Ottiene l'orario dell'ultima quotazione
Spread	Ottiene la quantità di spread (in punti)
SpreadFloat	Ottiene la flag di spread variabile
TicksBookDepth	Ottiene la profondità di ticks saving
Levels	

Controlling	
StopsLevel	Ottiene l'indentazione minima per gli ordini (in punti)
FreezeLevel	Ottiene la distanza delle operazioni di trade di congelamento (in punti)
Prezzi Bid	
Bid	Ottiene il prezzo di Bid corrente
BidHigh	Ottiene il prezzo massimo di Bid per il giorno
BidLow	Ottiene il prezzo minimo di Bid per il giorno
Prezzi Ask	
Ask	Ottiene il prezzo di Ask corrente
AskHigh	Ottiene il prezzo massimo di Ask per il giorno
AskLow	Ottiene il prezzo minimo di Ask per il giorno
Prices	
Last	Ottiene l'attuale ultimo prezzo (Last9
LastHigh	Ottiene il massimo ultimo prezzo (Last) per il giorno
LastLow	Ottiene il minimo ultimo prezzo (Last) per il giorno
Modalità di Trade	
TradeCalcMode	Ottiene la modalità di calcolo dei costi del contratto
TradeCalcModeDescription	Ottiene la modalità di calcolo dei costi del contratto, come stringa
TradeMode	Ottiene il tipo di esecuzione degli ordini
TradeModeDescription	Ottiene il tipo di esecuzione degli ordini, come stringa
TradeExecution	Ottiene la modalità di esecuzione dei trades
TradeExecutionDescription	Ottiene la modalità di esecuzione sottoforma di stringa
Swaps	
SwapMode	Ottiene la modalità di calcolo dello swap
SwapModeDescription	Ottiene la modalità di calcolo dello swap, come stringa
SwapRollover3days	Ottiene il giorno del triplo carico dello swap
SwapRollover3daysDescription	Ottiene il giorno del triplo carico dello swap, come stringa
Margini e flags	
MarginInitial	Ottiene il valore del margine iniziale
MarginMaintenance	Ottiene il valore del margine di mantenimento
MarginLong	Ottiene il tasso di carico del margine per le posizioni long

Controlling	
MarginShort	Ottiene il tasso di carico del margine per le posizioni short
MarginLimit	Ottiene il tasso di carico del margine per gli ordini Limit
MarginStop	Ottiene il tasso di carico del margine per gli ordini di Stop
MarginStopLimit	Ottiene il tasso di carico del margine per gli ordini StopLimit
TradeTimeFlags	Ottiene le flags delle modalità di scadenza consentite
TradeFillFlags	Ottiene le flags delle modalità di riempimento consentite
Quantizzazione	
Digits	Ottiene il numero di cifre dopo il periodo
Point	Ottiene il valore di un punto
TickValue	Ottiene il valore del tick (variazione minima di prezzo)
TickValueProfit	Ottiene il prezzo tick calcolato per una posizione profittevole
TickValueLoss	Ottiene il prezzo tick calcolato per una posizione in perdita
TickSize	Ottiene la variazione minima di prezzo
Grandezze dei contratti	
ContractSize	Ottiene la grandezza del contratto di trade
LotsMin	Ottiene il volume minimo per chiudere un affare
LotsMax	Ottiene il volume massimo per chiudere un affare
LotsStep	Ottiene il passo minimo di variazione di volume per chiudere un affare
LotsLimit	Ottiene il volume massimo consentito della posizione aperta ed ordini pendenti (direction insensitive) per un simbolo
Grandezze degli swap	
SwapLong	Ottiene il valore dello swap della posizione long
SwapShort	Ottiene il valore dello swap della posizione short
Proprietà del testo	
CurrencyBase	Ottiene il nome della valuta del simbolo base
CurrencyProfit	Ottiene il nome della valuta di profitto
CurrencyMargin	Ottiene il nome della valuta del margine
Bank	Ottiene il nome della fonte di quotazione corrente
Descrizione	Ottiene la stringa di descrizione del simbolo
Path	Ottiene il percorso in albero simboli

Controlling	
Proprietà del simbolo	
SessionDeals	Ottiene il numero di deals nella sessione corrente
SessionBuyOrders	Ottiene il numero di ordini di Buy al momento
SessionSellOrders	Ottiene il numero di ordini di Sell al momento
SessionTurnover	Ottiene la sintesi del turnover della sessione corrente
SessionInterest	Ottiene la sintesi di open interest della sessione corrente
SessionBuyOrdersVolume	Ottiene l'attuale volume di ordini di Buy
SessionSellOrdersVolume	Ottiene l'attuale volume di ordini di Sell
SessionOpen	Ottiene il prezzo di apertura della sessione corrente
SessionClose	Ottiene il prezzo di chiusura della sessione corrente
SessionAW	Ottiene il prezzo medio ponderato della sessione corrente
SessionPriceSettlement	Ottiene il prezzo di liquidazione della sessione corrente
SessionPriceLimitMin	Ottiene il prezzo minimo della sessione corrente
SessionPriceLimitMax	Ottiene il prezzo massimo della sessione corrente
L'accesso alle funzioni API MQL5	
InfoInteger	Ottiene il valore della proprietà di tipo integer specificata
InfoDouble	Ottiene il valore della proprietà di tipo double specificata
InfoString	Ottiene il valore della proprietà di tipo stringa specificata
Funzioni di servizio	
NormalizePrice	Restituisce il valore del prezzo, normalizzato utilizzando le proprietà del simbolo

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Refresh

Aggiorna i dati dei simboli.

```
void  Refresh()
```

Valore di ritorno

Nessuno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

RefreshRates

Aggiorna i dati delle quotazioni del simbolo.

```
bool  RefreshRates()
```

Valore di ritorno

true - successo, false - impossibile aggiornare le quotazioni.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Name

Ottiene il nome del simbolo.

```
string  Name() const
```

Valore di ritorno

Nome del simbolo.

Name

Imposta il nome del simbolo.

```
bool  Name(string name)
```

Valore di ritorno

Nessuno.

Select

Ottiene la flag simbolo "Market Watch".

```
bool  Select() const
```

Valore di ritorno

"Market Watch" flag del simbolo.

Select

Imposta la flag simbolo "Market Watch".

```
bool  Select()
```

Valore di ritorno

true - successo, false - impossibile cambiare la flag.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

IsSynchronized

Controlla la sincronizzazione del simbolo con il server.

```
bool  IsSynchronized() const
```

Valore di ritorno

true - se il simbolo è sincronizzato con il server, false - se non lo è.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Volume

Ottiene il volume dello scorso affare.

```
long  Volume() const
```

Valore di ritorno

Volume dell' ultimo affare.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

VolumeHigh

Ottiene il volume massimo della giornata.

```
long  VolumeHigh() const
```

Valore di ritorno

Volume massimo del giorno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

VolumeLow

Ottiene il volume minimo della giornata.

```
long  VolumeLow() const
```

Valore di ritorno

Volume minimo della giornata.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Time

Ottiene l'orario dell'ultima quotazione.

```
datetime  Time() const
```

Valore di ritorno

Orario dell'ultima quotazione.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Spread

Ottiene la quantità di spread (in punti).

```
int  Spread() const
```

Valore di ritorno

Ottiene la quantità di spread (in punti).

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SpreadFloat

Ottiene la flag di spread variabile.

```
bool  SpreadFloat() const
```

Valore di ritorno

Flag di spread variabile.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TicksBookDepth

Ottiene la profondità di salvataggio dei ticks.

```
int  TicksBookDepth() const
```

Valore di ritorno

Profondità di salvataggio dei ticks.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

StopsLevel

Ottiene il livello minimo dei livelli di stop per gli ordini (in punti).

```
int  StopsLevel() const
```

Valore di ritorno

Livello minimo dei livelli di stop per gli ordini (in punti).

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

FreezeLevel

Ottiene il livello di congelamento (in punti).

```
int  FreezeLevel() const
```

Valore di ritorno

Distanza del livello di congelamento (in punti).

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Bid

Ottiene il prezzo del Bid corrente.

```
double  Bid() const
```

Valore di ritorno

Prezzo Bid corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

BidHigh

Ottiene il prezzo massimo di Bid per il giorno

```
double  BidHigh() const
```

Valore di ritorno

Massimo prezzo Bid del giorno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

BidLow

Ottiene il prezzo minimo di Bid per il giorno

```
double  BidLow() const
```

Valore di ritorno

Minimo prezzo Bid del giorno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Ask

Ottiene il prezzo Ask corrente.

```
double  Ask() const
```

Valore di ritorno

Corrente prezzo Ask.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

AskHigh

Ottiene il massimo prezzo Ask per il giorno.

```
double  AskHigh() const
```

Valore di ritorno

Massimo prezzo Ask del giorno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

AskLow

Ottiene il prezzo minimo di Ask per il giorno

```
double  AskLow() const
```

Valore di ritorno

Minimo prezzo Ask del giorno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Last

Ottiene l'attuale prezzo Last (ultimo prezzo).

```
double  Last() const
```

Valore di ritorno

Attuale ultimo prezzo (Last).

LastHigh

Ottiene il massimo ultimo prezzo(Last) del giorno.

```
double  LastHigh() const
```

Valore di ritorno

MAssimo ultimo prezzo (Last) del giorno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

LastLow

Ottiene il minimo ultimo prezzo del giorno.

```
double  LastLow() const
```

Valore di ritorno

Minimo ultimo prezzo del giorno.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeCalcMode

Ottiene la modalità di calcolo dei costi del contratto.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

Valore di ritorno

Modalità di calcolo del costo del contratto dall'enumerazione ENUM_SYMBOL_CALC_MODE.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeCalcModeDescription

Ottiene la modalità di calcolo dei costi del contratto come stringa.

```
string  TradeCalcModeDescription() const
```

Valore di ritorno

Modalità di calcolo dei costi del contratto come stringa.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeMode

Ottiene il tipo di esecuzione degli ordini.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

Valore di ritorno

Tipo di esecuzione degli ordini dall'enumerazione ENUM_SYMBOL_TRADE_MODE.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeModeDescription

Ottiene la modalità di trade sotto forma di stringa.

```
string  TradeModeDescription() const
```

Valore di ritorno

Modalità di trade, come stringa.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeExecution

Ottiene la modalità di esecuzione dei trade.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

Valore di ritorno

Modalità di esecuzione del trade dall'enumerazione ENUM_SYMBOL_TRADE_EXECUTION.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeExecutionDescription

Ottiene la descrizione della modalità di esecuzione dei trade come stringa.

```
string  TradeExecutionDescription() const
```

Valore di ritorno

Modalità di esecuzione del trade sottoforma di stringa

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SwapMode

Ottiene la modalità di calcolo dello swap.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

Valore di ritorno

Modalità di calcolo dello Swap dall'enumerazione ENUM_SYMBOL_SWAP_MODE.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SwapModeDescription

Ottiene la descrizione della modalità di swap sotto forma di stringa.

```
string  SwapModeDescription() const
```

Valore di ritorno

Descrizione della modalità di swap come stringa.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SwapRollover3days

Ottiene il giorno di swap rollover.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

Valore di ritorno

Giorno Swap rollover dall'enumerazione ENUM_DAY_OF_WEEK.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SwapRollover3daysDescription

Ottiene il giorno di swap rollover come stringa.

```
string  SwapRollover3daysDescription() const
```

Valore di ritorno

Il giorno di swap rollover come stringa.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

MarginInitial

Ottiene il valore del margine iniziale.

```
double  MarginInitial()
```

Valore di ritorno

Valore del margine iniziale.

Nota

Esso restituisce la quantità di margine (in valuta margine dello strumento) che viene caricata da un lotto. Usato per controllare l'equità del cliente, quando entra nel mercato.

Il simbolo dovrebbe essere selezionato col metodo Nome.

MarginMaintenance

Ottiene il valore del margine di mantenimento.

```
double  MarginMaintenance()
```

Valore di ritorno

Valore del margine di mantenimento.

Nota

Esso restituisce la quantità di margine (in valuta margine dello strumento) che viene caricata da un lotto. Usato per controllare equità del cliente, quando lo stato dell' account è cambiato. Se il margine di mantenimento è uguale a 0, allora viene utilizzato il margine iniziale.

Il simbolo dovrebbe essere selezionato col metodo Nome.

MarginLong

Ottiene il tasso di margine di ricarica su posizioni long.

```
double  MarginLong() const
```

Valore di ritorno

Tasso di margine di ricarica su posizioni long.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

MarginShort

Ottiene il tasso di margine di ricarica su posizioni short.

```
double  MarginShort() const
```

Valore di ritorno

Tasso di margine di ricarica su posizioni short.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

MarginLimit

Ottiene il tasso di margine di ricarica su ordini Limit.

```
double  MarginLimit() const
```

Valore di ritorno

Tasso di margine di ricarica su ordini Limit.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

MarginStop

Ottiene il tasso di margine di ricarica sugli ordini di Stop.

```
double  MarginStop() const
```

Valore di ritorno

Tasso di margine di ricarica su ordini Stop.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

MarginStopLimit

Ottiene il tasso di ricarica margine su ordini Stop Limit.

```
double  MarginStopLimit() const
```

Valore di ritorno

Tasso di margine di ricarica su ordini Stop Limit.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeTimeFlags

Ottiene i flag di modalità di scadenza consentiti.

```
int  TradeTimeFlags() const
```

Valore di ritorno

Flags della modalità di scadenza ammessa.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TradeFillFlags

Ottiene le flag di modalità di riempimento consentite.

```
int  TradeFillFlags() const
```

Valore di ritorno

Flags di modalità di riempimento consentite.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Digits

Ottiene il numero di cifre dopo il periodo.

```
int  Digits() const
```

Valore di ritorno

Il numero di cifre dopo il periodo.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Point

Ottiene il valore di un punto.

```
double  Point() const
```

Valore di ritorno

Valore di un punto.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TickValue

Ottiene il valore di tick (variazione minima di prezzo).

```
double  TickValue() const
```

Valore di ritorno

Valore Tick (variazione minima di prezzo).

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TickValueProfit

Ottiene il prezzo tick calcolato per una posizione profittevole.

```
double  TickValueProfit() const
```

Valore di ritorno

Prezzo tick calcolato per una posizione profittevole.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TickValueLoss

Ottiene il prezzo tick calcolato per una posizione in perdita.

```
double  TickValueLoss() const
```

Valore di ritorno

Prezzo tick calcolato per una posizione in perdita.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

TickSize

Ottiene la variazione minima di prezzo.

```
double  TickSize() const
```

Valore di ritorno

Variazione minima di prezzo.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

ContractSize

Ottiene la quantità del contratto di trade.

```
double  ContractSize() const
```

Valore di ritorno

Importo del contratto di trade.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

LotsMin

Ottiene il volume minimo per chiudere un affare.

```
double  LotsMin() const
```

Valore di ritorno

Volume minimo per chiudere un affare.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

LotsMax

Ottiene il volume massimo per chiudere un affare.

```
double  LotsMax() const
```

Valore di ritorno

Volume massimo per chiudere un affare.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

LotsStep

Ottiene lo step minimo di variazione di volume per chiudere un affare.

```
double  LotsStep() const
```

Valore di ritorno

Step minimo di variazione di volume per chiudere un affare.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

LotsLimit

Ottiene il massimo consentito volume della posizione aperta ed ordini pendenti (direzione insensitive) per un simbolo.

```
double  LotsLimit() const
```

Valore di ritorno

Il massimo volume consentito della posizione aperta ed ordini pendenti (direzione insensitive) per un simbolo.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SwapLong

Ottiene il valore dello swap della posizione long.

```
double  SwapLong() const
```

Valore di ritorno

Valore dello swap della posizione long.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SwapShort

Ottiene il valore dello swap della posizione short.

```
double  SwapShort() const
```

Valore di ritorno

Valore dello swap della posizione short.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

CurrencyBase

Ottiene il nome del simbolo della valuta di base.

```
string  CurrencyBase() const
```

Valore di ritorno

Nome del simbolo della valuta di base.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

CurrencyProfit

Ottiene il nome della valuta con profitto.

```
string  CurrencyProfit() const
```

Valore di ritorno

Nome valuta profitto

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

CurrencyMargin

Ottiene il nome della valuta del margine.

```
string  CurrencyMargin() const
```

Valore di ritorno

Nome della valuta Margine.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Bank

Ottiene il nome della fonte di quotazione corrente.

```
string  Bank() const
```

Valore di ritorno

Nome dell'attuale fonte di quotazione.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Descrizione

Ottiene la stringa di descrizione del simbolo.

```
string  Description() const
```

Valore di ritorno

Stringa di descrizione del simbolo.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

Path

Ottiene il percorso come albero di simboli.

```
string  Path() const
```

Valore di ritorno

Ottiene il percorso come albero di simboli.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionDeals

Ottiene il numero di affari nella sessione corrente.

```
long  SessionDeals() const
```

Valore di ritorno

Numero di affari nella sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionBuyOrders

Ottiene il numero di ordini di Buy al momento.

```
long  SessionBuyOrders() const
```

Valore di ritorno

Numero di ordini di Buy al momento.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionSellOrders

Ottiene poi il numero di ordini Sell in questo momento.

```
long  SessionSellOrders() const
```

Valore di ritorno

Numero di ordini Sell in questo momento.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionTurnover

Ottiene la sintesi del turnover della sessione corrente.

```
double  SessionTurnover() const
```

Valore di ritorno

Sintesi del turnover della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionInterest

Ottiene la sintesi di interesse aperto (open interest) della sessione corrente.

```
double  SessionInterest() const
```

Valore di ritorno

Sintesi di interesse aperto (open interest) della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionBuyOrdersVolume

Ottiene l'attuale volume di ordini di Buy.

```
double  SessionBuyOrdersVolume() const
```

Valore di ritorno

Corrente volume di ordini di Buy.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionSellOrdersVolume

Ottiene l'attuale volume di ordini di Sell.

```
double  SessionSellOrdersVolume() const
```

Valore di ritorno

Attuale volume di ordini di Sell.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionOpen

Ottiene il prezzo di apertura(open) della sessione corrente.

```
double  SessionOpen() const
```

Valore di ritorno

Prezzo open della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionClose

Ottiene il prezzo di chiusura(close) della sessione corrente.

```
double  SessionClose() const
```

Valore di ritorno

Prezzo close della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionAW

Ottiene il prezzo medio ponderato della sessione corrente.

```
double  SessionAW() const
```

Valore di ritorno

Prezzo medio ponderato(average weighted price) della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionPriceSettlement

Ottiene il prezzo di liquidazione della sessione corrente.

```
double  SessionPriceSettlement() const
```

Valore di ritorno

Prezzo di liquidazione(settlement) della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionPriceLimitMin

Ottiene il prezzo minimo della sessione corrente.

```
double  SessionPriceLimitMin() const
```

Valore di ritorno

Prezzo minimo della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

SessionPriceLimitMax

Ottiene il prezzo massimo della sessione corrente.

```
double  SessionPriceLimitMax() const
```

Valore di ritorno

Prezzo massimo della sessione corrente.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

InfoInteger

Ottiene il valore della specificata proprietà di tipo integer.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // ID proprietà
   long&                     var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID di tipo indeger della proprietà dall'enumerazione ENUM_SYMBOL_INFO_INTEGER.

var

[out] Riferimento alla variabile di tipo long per piazzare i risultati.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

InfoDouble

Ottiene il valore della proprietà di tipo double specificata.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // ID della proprietà
   double&                  var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di tipo double dall'enumerazione ENUM_SYMBOL_INFO_DOUBLE.

var

[out] Riferimento alla variabile di tipo double per piazzare i risultati.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

InfoString

Ottiene il valore della proprietà di tipo string, specificata.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // ID della proprietà
   string&                  var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà testo.

var

[out] Riferimento alla variabile di tipo string per piazzare il risultato.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

NormalizePrice

Restituisce il valore del prezzo normalizzato utilizzando le proprietà del simbolo.

```
double  NormalizePrice(
   double  price   // prezzo
   ) const
```

Parametri

price

[in] Prezzo.

Valore di ritorno

Prezzo normalizzato.

Nota

Il simbolo dovrebbe essere selezionato col metodo Nome.

COrderInfo

COrderInfo è una classe per il facile accesso alle proprietà degli ordini pendenti.

Descrizione

La Classe COrderInfo fornisce l'accesso alle proprietà degli ordini pendenti.

Dichiarazione

```
class COrderInfo : public CObject
```

Titolo

```
#include <Trade\OrderInfo.mqh>
```

Gerarchia di ereditarietà

CObject

COrderInfo

I metodi di classe per gruppi

L'accesso alla proprietà di tipo integer	
Ticket	Ottiene il ticket di un ordine, precedentemente selezionato per l'accesso
TimeSetup	Ottiene l'orario dell'effettuazione dell'ordine
TimeSetupMsc	Riceve l'orario di piazzamento dell'ordine in millisecondi dal 01.01.1970
OrderType	Ottiene il tipo di ordine
OrderTypeDescription	Ottiene il tipo di ordine come una stringa
State	Ottiene lo stato dell'ordine
StateDescription	Ottiene lo stato dell'ordine come una stringa
TimeExpiration	Ottiene l'orario della scadenza dell'ordine
TimeDone	Ottiene il tempo di esecuzione degli ordini o la cancellazione
TimeDoneMsc	Riceve l'orario di esecuzione degli ordini o l'orario della cancellazione in millisecondi dal 01.01.1970
TypeFilling	Ottiene il tipo di esecuzione degli ordini per rimanenza
TypeFillingDescription	Ottiene il tipo di esecuzione degli ordini per rimanenza, sottoforma di stringa
TypeTime	Ottiene il tipo di ordine al momento dell'espirazione

L'accesso alla proprietà di tipo integer	
TypeTimeDescription	Ottiene il tipo ordine per data di scadenza(espirazione), come stringa
Magic	Ottiene l'ID dell'expert che ha effettuato l'ordine
PositionId	Ottiene l'ID della posizione
L'accesso alla proprietà di tipo double	
VolumeInitial	Ottiene il volume iniziale dell' ordine
VolumeCurrent	Ottiene il volume non riempito dell' ordine
PriceOpen	Ottiene il prezzo dell'ordine
StopLoss	Ottiene lo Stop Loss dell'ordine
TakeProfit	Ottiene il Take Profit dell'ordine
PriceCurrent	Ottiene il prezzo corrente per simbolo dell' ordine
PriceStopLimit	Ottiene il prezzo di un ordine Limit
L'accesso alla proprietà testuali	
Symbol	Ottiene il nome del simbolo dell' ordine
Comment	Ottiene il commento dell' ordine
L'accesso alle funzioni API MQL5	
InfoInteger	Ottiene il valore della proprietà di tipo integer specificata
InfoDouble	Ottiene il valore della proprietà di tipo double specificata
InfoString	Ottiene il valore della proprietà di tipo string specificata
State	
StoreState	Salva i parametri dell'ordine
CheckState	Controlla i parametri correnti rispetto ai parametri salvati
Selezione	
Select	Seleziona un ordine per ticket, per ulteriore accesso alle sue proprietà
SelectByIndex	Seleziona un ordine per indice, per un ulteriore accesso alle sue proprietà

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ticket

Ottiene il ticket di un ordine.

```
ulong  Ticket() const
```

Valore di ritorno

Ticket dell' ordine in caso di successo, altrimenti - ULONG_MAX.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TimeSetup

Ottiene l'orario dell'effettuazione dell'ordine.

```
datetime  TimeSetup() const
```

Valore di ritorno

Orario di piazzamento dell'ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TimeSetupMsc

Riceve l'orario di piazzamento dell'ordine per l'esecuzione in millisecondi dal 01.01.1970.

```
ulong  TimeSetupMsc() const
```

Valore di ritorno

Il tempo di piazzamento di un ordine per l'esecuzione in millisecondi dal 01.01.1970.

Nota

L'ordine deve essere preliminarmente selezionato per l'accesso tramite i metodi Select (per ticket) o SelectByIndex (per indice).

OrderType

Ottiene il tipo di ordine.

```
ENUM_ORDER_TYPE  OrderType()
```

Valore di ritorno

Tipo di ordine dall'enumerazione ENUM_ORDER_TYPE.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TypeDescription

Ottiene il tipo di ordine, come stringa.

```
string  TypeDescription() const
```

Valore di ritorno

Tipo di ordine come stringa.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

State

Ottiene lo stato dell'ordine.

```
ENUM_ORDER_STATE  State() const
```

Valore di ritorno

Stato dell'ordine dall'enumerazione ENUM_ORDER_STATE.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

StateDescription

Ottiene lo stato dell'ordine come una stringa.

```
string  StateDescription() const
```

Valore di ritorno

Stato dell'ordine come una stringa.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TimeExpiration

Ottiene l'orario di espirazione(scadenza) dell' ordine.

```
datetime  TimeExpiration() const
```

Valore di ritorno

Orario di espirazione dell'ordine, impostato al suo piazzamento.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TimeDone

Ottiene l'orario di esecuzione o cancellazione dell'ordine.

```
datetime  TimeDone() const
```

Valore di ritorno

Orario di esecuzione o cancellazione dell'ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TimeDoneMsc

Riceve gli orari di esecuzione o cancellazione dell'ordine in millisecondi dal 01.01.1970.

```
ulong  TimeDoneMsc() const
```

Valore di ritorno

Orario di esecuzione dell'ordine oppure orario di cancellazione dell'ordine in millisecondi dal 01.01.1970.

Nota

L'ordine deve essere preliminarmente selezionato per l'accesso tramite i metodi Select (per ticket) o SelectByIndex (per indice).

TypeFilling

Ottiene il tipo di filling dell'ordine.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valore di ritorno

Tipo di riempimento dell'ordine dall'enumerazione ENUM_ORDER_TYPE_FILLING.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TypeFillingDescription

Ottiene il tipo di filling dell'ordine, come stringa.

```
string  TypeFillingDescription() const
```

Valore di ritorno

Tipo di riempimento dell'ordine, come stringa.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TypeTime

Ottiene il tipo di ordine al momento della scadenza.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valore di ritorno

Tipo di ordine al momento della scadenza dall'enumerazione ENUM_ORDER_TYPE_TIME.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TypeTimeDescription

Ottiene il tipo ordine per data di scadenza, come stringa.

```
string  TypeTimeDescription() const
```

Valore di ritorno

Tipo di ordine per data di scadenza, come stringa.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

Magic

Ottiene l'ID dell' Expert Advisor che ha effettuato l'ordine.

```
long  Magic() const
```

Valore di ritorno

ID dell' Expert Advisor che ha effettuato l'ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

PositionId

Ottiene l'ID della posizione.

```
long  PositionId() const
```

Valore di ritorno

ID della posizione in cui l'ordine era coinvolto.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

VolumeInitial

Ottiene il volume iniziale dell' ordine.

```
double  VolumeInitial() const
```

Valore di ritorno

Volume iniziale dell' ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

VolumeCurrent

Ottiene il volume non riempito dell' ordine.

```
double  VolumeCurrent() const
```

Valore di ritorno

il volume non riempito dell' ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

PriceOpen

Ottiene il prezzo dell'ordine.

```
double  PriceOpen() const
```

Valore di ritorno

Prezzo del conferimento dell'ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

StopLoss

Ottiene lo Stop Loss dell'ordine.

```
double  StopLoss() const
```

Valore di ritorno

Stop Loss dell'Ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

TakeProfit

Ottiene il Take Profit dell'ordine.

```
double  TakeProfit() const
```

Valore di ritorno

Take Profit dell'Ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

PriceCurrent

Ottiene il prezzo corrente dal simbolo dell'ordine.

```
double  PriceCurrent() const
```

Valore di ritorno

Prezzo corrente dal simbolo dell'ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

PriceStopLimit

Ottiene il prezzo di un ordine pendente.

```
double  PriceStopLimit() const
```

Valore di ritorno

Prezzo di impostazione di un ordine pendente.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

Symbol

Ottiene il nome del simbolo dell'ordine.

```
string  Symbol() const
```

Valore di ritorno

Nome del simbolo dell'ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

Comment

Ottiene il commento dell' ordine.

```
string  Comment() const
```

Valore di ritorno

Commento dell' ordine.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

InfoInteger

Ottiene il valore della specificata proprietà di tipo integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // ID proprietà
   long&                        var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di topo integer dall'enumerazione ENUM_ORDER_PROPERTY_INTEGER.

var

[out] Riferimento alla variabile di tipo long per piazzare i risultati.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

InfoDouble

Ottiene il valore della proprietà di tipo double specificata.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // ID proprietà
   double&                     var          // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di tipo double dall'enumerazione ENUM_ORDER_PROPERTY_DOUBLE.

var

[out] Riferimento alla variabile di tipo double per piazzare i risultati.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

InfoString

Ottiene il valore della proprietà di tipo string, specificata.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // ID proprietà
   string&                     var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà stringa dall'enumerazione ENUM_ORDER_PROPERTY_STRING.

var

[out] Riferimento alla variabile di tipo string per piazzare il risultato.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

L'ordine deve essere selezionato usando i metodi Select (da ticket) o SelectByIndex (da indice).

StoreState

Salva i parametri dell'ordine.

```
void  StoreState()
```

Valore di ritorno

Nessuno.

CheckState

Controlla i parametri correnti rispetto ai parametri salvati.

```
bool  CheckState()
```

Valore di ritorno

true - se i parametri dell'ordine sono cambiati dopo l'ultima chiamata del metodo StoreState(), altrimenti - false.

Select

Seleziona un ordine da ticket per ulteriore accesso alle sue proprietà.

```
bool  Select(
   ulong  ticket   // ticket dell'ordine
   )
```

Valore di ritorno

true - successo, false - impossibile selezionare l'ordine.

SelectByIndex

Seleziona un ordine in base all'indice.

```
bool  SelectByIndex(
   int  index   // indice ordine
   )
```

Parametri

index

[in] Indice dell'ordine.

Valore di ritorno

true - successo, false - impossibile selezionare l'ordine.

CHistoryOrderInfo

CHistoryOrderInfo è una classe per il facile accesso alle proprietà dello storico ordini.

Descrizione

La Classe CHistoryOrderInfo offre un facile accesso alle proprietà dello storico degli ordini.

Dichiarazione

```
class CHistoryOrderInfo : public CObject
```

Titolo

```
#include <Trade\HistoryOrderInfo.mqh>
```

Gerarchia di ereditarietà

CObject

CHistoryOrderInfo

I metodi di classe per gruppi

L'accesso alla proprietà di tipo integer	
TimeSetup	Ottiene l'orario dell'effettuazione dell'ordine
TimeSetupMsc	Riceve l'orario di piazzamento di un ordine in millisecondi dal 01.01.1970
OrderType	Ottiene il tipo di ordine
OrderTypeDescription	Ottiene il tipo di ordine come una stringa
State	Ottiene lo stato dell'ordine
StateDescription	Ottiene lo stato dell'ordine come una stringa
TimeExpiration	Ottiene l'orario della scadenza dell'ordine
TimeDone	Ottiene il tempo di esecuzione degli ordini o la cancellazione
TimeDoneMsc	Riceve l'orario di esecuzione degli ordini o l'orario della cancellazione in millisecondi dal 01.01.1970
TypeFilling	Ottiene il tipo di esecuzione degli ordini per rimanenza
TypeFillingDescription	Ottiene il tipo di esecuzione degli ordini per rimanenza, sottoforma di stringa
TypeTime	Ottiene il tipo di ordine al momento dell'espirazione
TypeTimeDescription	Ottiene il tipo ordine per data di scadenza(espirazione), come stringa

L'accesso alla proprietà di tipo integer	
Magic	Ottiene l'ID dell'expert che ha effettuato l'ordine
PositionId	Ottiene l'ID della posizione
L'accesso alla proprietà di tipo double	
VolumeInitial	Ottiene il volume iniziale dell' ordine
VolumeCurrent	Ottiene il volume non riempito dell' ordine
PriceOpen	Ottiene il prezzo dell'ordine
StopLoss	Ottiene lo Stop Loss dell'ordine
TakeProfit	Ottiene il Take Profit dell'ordine
PriceCurrent	Ottiene il prezzo corrente per simbolo dell' ordine
PriceStopLimit	Ottiene il prezzo di un ordine Limit
L'accesso alla proprietà testuali	
Symbol	Ottiene il simbolo dell'ordine
Comment	Ottiene il commento dell' ordine
L'accesso alle funzioni API MQL5	
InfoInteger	Ottiene il valore della proprietà di tipo integer specificata
InfoDouble	Ottiene il valore della proprietà di tipo double specificata
InfoString	Ottiene il valore della proprietà di tipo string specificata
Selezione	
Ticket	Ottiene il ticket/seleziona l'ordine
SelectByIndex	Seleziona l'ordine in base all'indice

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

TimeSetup

Ottiene l'orario dell'effettuazione dell'ordine.

```
datetime  TimeSetup() const
```

Valore di ritorno

Orario di piazzamento dell'ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TimeSetupMsc

Riceve l'orario di piazzamento di un ordine per l'esecuzione, in millisecondi dal 01.01.1970.

```
ulong  TimeSetupMsc() const
```

Valore di ritorno

Il tempo di piazzamento di un ordine per l'esecuzione, in millisecondi dal 01.01.1970.

Nota

L' ordine storico deve essere preliminarmente selezionato per l'accesso tramite il metodo Ticket (per ticket) o SelectByIndex (per indice).

OrderType

Ottiene il tipo di ordine.

```
ENUM_ORDER_TYPE  OrderType() const
```

Valore di ritorno

Tipo di ordine dall'enumerazione ENUM_ORDER_TYPE.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TypeDescription

Ottiene il tipo di ordine, come stringa.

```
string  TypeDescription() const
```

Valore di ritorno

Tipo di ordine come stringa.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

State

Ottiene lo stato dell'ordine.

```
ENUM_ORDER_STATE  State() const
```

Valore di ritorno

Stato dell'ordine dall'enumerazione ENUM_ORDER_STATE.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

StateDescription

Ottiene lo stato dell'ordine come una stringa.

```
string  StateDescription() const
```

Valore di ritorno

Stato dell'ordine come una stringa.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TimeExpiration

Ottiene l'orario dell'espirazione dell'ordine.

```
datetime  TimeExpiration() const
```

Valore di ritorno

Orario di espirazione dell'ordine, impostato al suo piazzamento.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TimeDone

Ottiene l'orario di esecuzione o cancellazione dell'ordine.

```
datetime  TimeDone() const
```

Valore di ritorno

Orario di esecuzione o cancellazione dell'ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TimeDoneMsc

Riceve orario di esecuzione degli ordini oppure orario della cancellazione in millisecondi dal 01.01.1970.

```
ulong  TimeDoneMsc() const
```

Valore di ritorno

Orario di esecuzione o cancellazione dell'ordine in milliseconds since 01.01.1970.

Nota

L' ordine storico deve essere preliminarmente selezionato per l'accesso tramite il metodo Ticket (per ticket) o SelectByIndex (per indice).

TypeFilling

Ottiene il tipo di esecuzione dell' ordine per rimanenza.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valore di ritorno

Tipo di esecuzione dell'ordine da rimenenza dall'enumerazione ENUM_ORDER_TYPE_FILLING.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TypeFillingDescription

Ottiene il tipo di esecuzione dell' ordine per rimanenza, sottoforma di stringa.

```
string  TypeFillingDescription() const
```

Valore di ritorno

Tipo di esecuzione dell'ordine per rimenenza, sottoforma di stringa.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TypeTime

Ottiene il tipo di ordine al momento della scadenza.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valore di ritorno

Tipo di ordine al momento della scadenza dall'enumerazione ENUM_ORDER_TYPE_TIME.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TypeTimeDescription

Ottiene il tipo ordine per data di scadenza, come stringa.

```
string  TypeTimeDescription() const
```

Valore di ritorno

Tipo di ordine per data di scadenza, come stringa.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

Magic

Ottiene l'ID dell' Expert Advisor che ha effettuato l'ordine.

```
long  Magic() const
```

Valore di ritorno

ID dell' Expert Advisor che ha effettuato l'ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

PositionId

Ottiene l'ID della posizione.

```
long  PositionId() const
```

Valore di ritorno

ID della posizione in cui l'ordine era coinvolto.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

VolumeInitial

Ottiene il volume iniziale dell' ordine.

```
double  VolumeInitial() const
```

Valore di ritorno

Volume iniziale dell' ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

VolumeCurrent

Ottiene il volume non riempito dell' ordine.

```
double  VolumeCurrent() const
```

Valore di ritorno

il volume non riempito dell' ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

PriceOpen

Ottiene il prezzo dell'ordine.

```
double  PriceOpen() const
```

Valore di ritorno

Prezzo del conferimento dell'ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

StopLoss

Ottiene il prezzo di Stop Loss dell'ordine.

```
double  StopLoss() const
```

Valore di ritorno

Prezzo di Stop Loss dell'ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

TakeProfit

Ottiene il prezzo Take Profit dell'ordine.

```
double  TakeProfit() const
```

Valore di ritorno

Il prezzo Take Profit dell'ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

PriceCurrent

Ottiene il prezzo corrente del simbolo dell'ordine.

```
double  PriceCurrent() const
```

Valore di ritorno

Il prezzo attuale del simbolo dell' ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

PriceStopLimit

Ottiene il prezzo dell'ordine pendente.

```
double  PriceStopLimit() const
```

Valore di ritorno

Prezzo dell'ordine pendente.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

Symbol

Ottiene il nome del simbolo dell'ordine.

```
string  Symbol() const
```

Valore di ritorno

Nome del simbolo dell'ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

Comment

Ottiene il commento dell' ordine.

```
string  Comment() const
```

Valore di ritorno

Commento dell' ordine.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

InfoInteger

Ottiene il valore della specificata proprietà di tipo integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // ID proprietà
   long&                        var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di topo integer dall'enumerazione ENUM_ORDER_PROPERTY_INTEGER.

var

[out] Riferimento alla variabile di tipo long per piazzare i risultati.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

InfoDouble

Ottiene il valore della proprietà di tipo double specificata.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // ID proprietà
   double&                     var          // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di tipo double dall'enumerazione ENUM_ORDER_PROPERTY_DOUBLE.

var

[out] Riferimento alla variabile di tipo double per piazzare i risultati.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

InfoString

Ottiene il valore della proprietà di tipo string, specificata.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // ID proprietà
   string&                     var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà testo dall'enumerazione ENUM_ORDER_PROPERTY_STRING.

var

[out] Riferimento alla variabile di tipo string per piazzare il risultato.

Valore di ritorno

true - successo, false - incapace di ottenere i valori della proprietà.

Nota

L'ordine storico deve essere selezionato usando i metodi Ticket (per ticket) o SelectByIndex (per indice).

Ticket (Metodo Get)

Ottiene il ticket dell'ordine.

```
ulong  Ticket() const
```

Valore di ritorno

Ticket dell'ordine.

Ticket (Metodo Set)

Seleziona l'ordine per ulteriori lavori.

```
void  Ticket(
   ulong  ticket      // ticket
   )
```

Parametri

ticket

[in] Ticket ordine.

SelectByIndex

Seleziona un ordine storico per indice.

```
bool  SelectByIndex(
   int  index   // indice ordine
   )
```

Parametri

index

[in] Indice per ordine storico.

Valore di ritorno

true - successo, false - impossibile selezionare l'ordine.

CPositionInfo

CPositionInfo è una classe per il facile accesso alle proprietà della posizione aperta.

Descrizione

La Classe CPositionInfo offre un facile accesso alle proprietà della posizione aperta.

Dichiarazione

```
class CPositionInfo : public CObject
```

Titolo

```
#include <Trade\PositionInfo.mqh>
```

Gerarchia di ereditarietà

CObject

CPositionInfo

I metodi di classe per gruppi

L'accesso alla proprietà di tipo integer	
Time	Ottiene l'orario di apertura della posizione
TimeMsc	Riceve l'orario di apertura della posizione in millisecondi dal 01.01.1970
TimeUpdate	Riceve l'orario di cambiamento della posizione in secondi dal 01.01.1970
TimeUpdateMsc	Riceve l'orario di cambiamento della posizione in millisecondi dal 01.01.1970
PositionType	Ottiene il tipo di posizione
TypeDescription	Ottiene il tipo di posizione come stringa
Magic	Ottiene l'ID dell'expert, che ha aperto la posizione
Identifier	Ottiene l'ID della posizione
L'accesso alla proprietà di tipo double	
Volume	Ottiene il volume della posizione
PriceOpen	Ottiene il prezzo di apertura della posizione
StopLoss	Ottiene il prezzo di Stop Loss della posizione
TakeProfit	Ottiene il prezzo di Take Profit della posizione

L'accesso alla proprietà di tipo integer	
PriceCurrent	Ottiene il prezzo corrente dal simbolo della posizione
Commission	Ottiene l'importo della commissione dalla posizione
Swap	Ottiene la quantità di swap dalla posizione
Profit	Ottiene la quantità di profitto corrente dalla posizione
L'accesso alla proprietà testuali	
Symbol	Ottiene il nome del simbolo dalla posizione
Comment	Ottiene il commento della posizione
L'accesso alle funzioni API MQL5	
InfoInteger	Ottiene il valore della proprietà di tipo integer specificata
InfoDouble	Ottiene il valore della proprietà di tipo double specificata
InfoString	Ottiene il valore della proprietà di tipo stringa specificata
Selezione	
Select	Seleziona la posizione
SelectByIndex	Seleziona la posizione in base all'indice
SelectByMagic	Seleziona una posizione con il nome del simbolo e numero magico specificati
SelectByTicket	Seleziona la posizione da ticket
State	
StoreState	Salva i parametri della posizione
CheckState	Controlla i parametri correnti rispetto ai parametri salvati

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Time

Ottiene l'orario di apertura della posizione.

```
datetime  Time() const
```

Valore di ritorno

Orario di apertura della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex(by index) metodi.

TimeMsc

Riceve l'orario di apertura della posizione in millisecondi dal 01.01.1970.

```
ulong  TimeMsc() const
```

Valore di ritorno

Orario di apertura della Posizione in millisecondi dal 01.01.1970.

Nota

La posizione dovrebbe essere preliminarmente selezionata per l'accesso tramite il metodo Select (by symbol) o SelectByIndex (per indice).

TimeUpdate

Riceve l'orario di cambiamento della posizione in secondi dal 01.01.1970.

```
datetime  TimeUpdate() const
```

Valore di ritorno

Orario di cambiamento della posizione in secondi dal 01.01.1970.

Nota

La posizione dovrebbe essere preliminarmente selezionata per l'accesso tramite il metodo Select (by symbol) o SelectByIndex (per indice).

TimeUpdateMsc

Riceve l'orario di cambiamento della posizione in millisecondi dal 01.01.1970.

```
ulong  TimeUpdateMsc() const
```

Valore di ritorno

L'orario cambiamento della posizione in millisecondi dal 01.01.1970.

Nota

La posizione dovrebbe essere preliminarmente selezionata per l'accesso tramite il metodo Select (by symbol) o SelectByIndex (per indice).

PositionType

Ottiene il tipo di posizione.

```
ENUM_POSITION_TYPE  PositionType() const
```

Valore di ritorno

Tipo di posizione (valore dell'enumerazione ENUM_POSITION_TYPE).

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

TypeDescription

Ottiene il tipo di posizione, come stringa.

```
string  TypeDescription() const
```

Valore di ritorno

Tipo di posizione, come stringa.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Magic

Ottiene l'ID dell' Expert Advisor che ha aperto la posizione.

```
long  Magic() const
```

Valore di ritorno

ID dell' Expert Advisor che ha aperto la posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Identifier

Ottiene l'ID della posizione.

```
long  Identifier() const
```

Valore di ritorno

ID della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Volume

Ottiene il volume della posizione.

```
double  Volume() const
```

Valore di ritorno

Volume della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

PriceOpen

Ottiene il prezzo di apertura della posizione.

```
double  PriceOpen() const
```

Valore di ritorno

Prezzo di apertura della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

StopLoss

Ottiene il prezzo Stop Loss della posizione.

```
double  StopLoss() const
```

Valore di ritorno

Il prezzo Stop Loss della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

TakeProfit

Ottiene il prezzo Take Profit della posizione.

```
double  TakeProfit() const
```

Valore di ritorno

Il prezzo Take Profit della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

PriceCurrent

Ottiene il prezzo corrente per simbolo della posizione.

```
double  PriceCurrent() const
```

Valore di ritorno

Prezzo corrente dal simbolo della posizione.

Commission

Ottiene l'importo della commissione della posizione.

```
double  Commission() const
```

Valore di ritorno

Importo della commissione della posizione (in valuta di deposito).

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Swap

Ottiene la quantità di swap della posizione.

```
double  Swap() const
```

Valore di ritorno

Quantità di swap della posizione (in valuta di deposito).

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Profit

Ottiene la quantità del corrente profitto della posizione.

```
double  Profit() const
```

Valore di ritorno

Quantità di profitto corrente della posizione (in valuta di deposito).

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Symbol

Ottiene il nome del simbolo della posizione.

```
string  Symbol() const
```

Valore di ritorno

Nome del simbolo della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Comment

Ottiene il commento della posizione.

```
string  Comment() const
```

Valore di ritorno

Commento della posizione.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

InfoInteger

Ottiene il valore della specificata proprietà di tipo integer.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,      // ID della proprietà
   long&                           var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà tipo integer (valore dell'enumerazione ENUM_POSITION_PROPERTY_INTEGER).

var

[out] Riferimento alla variabile di tipo long per piazare il risultato.

Valore di ritorno

true - in caso di successo, false - se impossibile ottenere il valore della proprietà.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

InfoDouble

Ottiene il valore della proprietà di tipo double specificata.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // ID della proprietà
   double&                        var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di tipo double (valore dell'enumerazione ENUM_POSITION_PROPERTY_DOUBLE).

var

[in] Riferimento alla variabile di tipo double per piazzare il risultato.

Valore di ritorno

true - in caso di successo, false - se impossibile ottenere il valore della proprietà.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

InfoString

Ottiene il valore della proprietà di tipo string, specificata.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // ID della proprietà
   string&                        var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà testuale (valore dell'enumerazione ENUM_POSITION_PROPERTY_STRING).

var

[out] Riferimento alla variabile di tipo stringa per piazzare il risultato.

Valore di ritorno

true - in caso di successo, false - se impossibile ottenere il valore della proprietà.

Nota

La posizione deve essere selezionata con il metodo Select (by ticket) o SelectByIndex (per indice).

Select

Seleziona la posizione per ulteriori lavori.

```
bool  Select(
   const string  symbol      // simbolo
   )
```

Parametri

symbol

[in] Simbolo per la selezione della posizione.

SelectByIndex

Seleziona la posizione per indice per un ulteriore accesso alle sue proprietà.

```
bool  SelectByIndex(
   int  index   // indice posizione
   );
```

Valore di ritorno

true - in caso di successo, false - se non è possibile selezionare la posizione.

SelectByMagic

Seleziona una posizione in base al nome di uno strumento finanziario e numero magico, per l'ulteriore lavoro su di essa.

```
bool  SelectByMagic(
   const string  symbol, // Nome del simbolo
   const ulong   magic   // Numero magico
   );
```

Parametri

symbol

[in] Nome del simbolo.

magic

[in] Numero magico della posizione.

Valore di ritorno

Restituisce true in caso di successo o false se la posizione selezionata "non è riuscita".

SelectByTicket

Seleziona la posizione per ticket per ulteriori operazioni.

```
bool  SelectByTicket(
   ulong  ticket      // ticket della posizione
   );
```

Parametri

ticket

[in] Ticket della posizione.

Return value

true - successo, false - nessuna posizione selezionata .

StoreState

Salva i parametri della posizione.

```
void  StoreState()
```

Valore di ritorno

Nessuno.

CheckState

Controlla i parametri correnti rispetto ai parametri salvati.

```
bool  CheckState()
```

Valore di ritorno

true - se i parametri della posizione sono stati modificati dopo l'ultima chiamata del metoto StoreState(), altrimenti - false.

CDealInfo

CDealInfo è una classe per un facile accesso alle proprietà dell'affare(deal).

Descrizione

La Classe CDealInfo fornisce l'accesso alle proprietà dell' affare.

Dichiarazione

```
class CDealInfo : public CObject
```

Titolo

```
#include <Trade\DealInfo.mqh>
```

Gerarchia di ereditarietà

CObject

CDealInfo

I metodi di classe per gruppi

L'accesso alla proprietà di tipo integer	
Order	Ottiene il numero dell'ordine per cui viene eseguito l'affare
Time	Ottiene l'orario di esecuzione dell'affare
TimeMsc	Riceve l'orario di esecuzione dell'affare in millisecondi dal 01.01.1970
DealType	Ottiene il tipo dell'affare
TypeDescription	Ottiene il tipo d' affare, come una stringa
Entry	Ottiene la direzione dell'affare
EntryDescription	Ottiene la direzione dell'affare, come stringa
Magic	Ottiene l'ID dell' expert, che ha eseguito l'affare
PositionId	Ottiene l'ID della posizione, in cui è stato coinvolto l'affare
L'accesso alla proprietà di tipo double	
Volume	Ottiene il volume dell' affare
Price	Ottiene il prezzo dell'affare
Commision	Ottiene l'importo della commissione dell'affare
Swap	Ottiene la quantità di swap quando la posizione è chiusa
Profit	Ottiene il risultato finanziario dell'affare

L'accesso alla proprietà di tipo integer	
L'accesso alla proprietà testuali	
Symbol	Ottiene il nome del simbolo dell'affare
Comment	Ottiene il commento dell'affare
L'accesso alle funzioni API MQL5	
InfoInteger	Ottiene il valore della proprietà di tipo integer specificata
InfoDouble	Ottiene il valore della proprietà di tipo double specificata
InfoString	Ottiene il valore della proprietà di tipo string specificata
Selezione	
Ticket	Ottiene il ticket/seleziona, l'affare
SelectByIndex	Seleziona l'affare per indice

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Order

Ottiene in numero dell'ordine per cui viene eseguito l'affare.

```
long  Order() const
```

Valore di ritorno

Numero dell'ordine per cui viene eseguito l'affare.

Nota

L'affare dev'essere selezionato utilizzando i metodi Ticket (by ticket) o SelectByIndex (by index).

Time

Ottiene l'orario di esecuzione dell'affare.

```
datetime  Time() const
```

Valore di ritorno

Orario di esecuzione dell'affare.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

TimeMsc

Riceve l'orario dell' esecuzione dell'affare in millisecondi dal 01.01.1970.

```
ulong  TimeMsc() const
```

Valore di ritorno

L'orario di esecuzione dell'affare in millisecondi dal 01.01.1970.

Nota

L'affare dovrebbe essere preliminarmente selezionato per l'accesso tramite il metodo Ticket (per ticket) o SelectByIndex (per indice).

DealType

Ottiene il tipo dell'affare.

```
ENUM_DEAL_TYPE  DealType() const
```

Valore di ritorno

Tipo dell'affare (valore dell'enumerazione ENUM_DEAL_TYPE).

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

TypeDescription

Ottiene il tipo di affare come stringa.

```
string  TypeDescription() const
```

Valore di ritorno

Tipo di affare come stringa.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Entry

Ottiene la direzione dell'affare.

```
ENUM_DEAL_ENTRY  Entry() const
```

Valore di ritorno

Direzione del deal (valore dell'enumerazione ENUM_DEAL_ENTRY).

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

EntryDescription

Ottiene la direzione dell'affare, come una stringa.

```
string  EntryDescription() const
```

Valore di ritorno

Direzione del deal, come stringa.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Magic

Ottiene l'ID dell'Expert Advisor, che ha eseguito l'affare.

```
long  Magic() const
```

Valore di ritorno

ID dell' Expert Advisor, che ha eseguito l'affare.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

PositionId

Ottiene l'ID della posizione, in cui è stato coinvolto l'affare.

```
long  PositionId() const
```

Valore di ritorno

ID della posizione, in cui è stato coinvolto l'affare.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Volume

Ottiene il volume dell'affare.

```
double  Volume() const
```

Valore di ritorno

Volume dell' affare.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Price

Ottiene il prezzo dell'affare.

```
double  Price() const
```

Valore di ritorno

Prezzo dell'affare

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Commission

Ottiene l'importo della commissione dell'affare.

```
double  Commission() const
```

Valore di ritorno

Importo della commissione dell'affare.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Swap

Ottiene la quantità di swap quando la posizione viene chiusa.

```
double  Swap() const
```

Valore di ritorno

Quantità di swap quando la posizione viene chiusa.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Profit

Ottiene il risultato finanziario dell'affare.

```
double  Profit() const
```

Valore di ritorno

Risultato finanziario dell'affare (in valuta di deposito).

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Symbol

Ottiene il nome del simbolo dell' affare.

```
string  Symbol() const
```

Valore di ritorno

Nome del simbolo dell' affare.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Comment

Ottiene il commento dell'affare.

```
string  Comment() const
```

Valore di ritorno

Commento del deal.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

InfoInteger

Ottiene il valore della specificata proprietà di tipo integer.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,      // ID della proprietà
   long&                       var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di tipo integer (valore dell'enumerazione ENUM_DEAL_PROPERTY_INTEGER).

var

[out] Riferimento alla variabile di tipo long per piazare il risultato.

Valore di ritorno

true - in caso di successo, false - se impossibile ottenere il valore della proprietà.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

InfoDouble

Ottiene il valore della proprietà di tipo double specificata.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // ID della proprietà
   double&                    var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà di tipo double (valore dell'enumerazione ENUM_DEAL_PROPERTY_DOUBLE).

var

[out] Riferimento alla variabile di tipo double per piazzare il risultato.

Valore di ritorno

true - in caso di successo, false - se impossibile ottenere il valore della proprietà.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

InfoString

Ottiene il valore della proprietà di tipo string, specificata.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // ID della proprietà
   string&                    var           // riferimento alla variabile
   ) const
```

Parametri

prop_id

[in] ID della proprietà testuale (valore dell'enumerazione ENUM_DEAL_PROPERTY_STRING).

var

[out] Riferimento alla variabile di tipo stringa per piazzare il risultato.

Valore di ritorno

true - in caso di successo, false - se impossibile ottenere il valore della proprietà.

Nota

L'affare dovrebbe essere selezionato con i metodi Ticket (per ticket) o SelectByIndex (per indice).

Ticket (Metodo Get)

Ottiene il ticket.

```
ulong  Ticket() const
```

Valore di ritorno

Ticket dell'affare.

Ticket (Metodo Set)

Seleziona la posizione per ulteriori lavori.

```
void  Ticket(
   ulong  ticket      // ticket
   )
```

Parametri

ticket

[in] Ticket dell'affare.

SelectByIndex

Seleziona l'affare per indice per ulteriore accesso alle sue proprietà.

```
bool  SelectByIndex(
   int  index   // indice ordine
   )
```

Valore di ritorno

true - in caso di successo, false - se non è possibile selezionare l'affare.

CTrade

CTrade è una classe per un facile accesso alle funzioni di trading.

Descrizione

La classe CTrade offre un facile accesso alle funzioni di trade.

Dichiarazione

```
class CTrade : public CObject
```

Titolo

```
#include <Trade\Trade.mqh>
```

Gerarchia di ereditarietà

CObject

CTrade

Discendenti diretti

CExpertTrade

I metodi di classe per gruppi

Setting parameters	
LogLevel	Imposta il livello di logging
SetExpertMagicNumber	Imposta l' expert ID
SetDeviationInPoints	Imposta la deviazione consentita
SetTypeFilling	Imposta il tipo di riempimento dell' ordine
SetTypeFillingBySymbol	Imposta il tipo di il riempimento (filling) dell'ordine in base alle impostazioni del simbolo specificato
SetAsyncMode	Imposta la modalità asincrona per le operazioni di trading
SetMarginMode	Imposta la modalità di calcolo del margine secondo le impostazioni dell'account
Operazioni con gli ordini	
OrderOpen	Piazza un ordine pendente con i parametri specificati
OrderModify	Modifica i parametri d'ordine pendente
OrderDelete	Elimina un ordine pendente
Operazioni con le posizioni	
PositionOpen	Si apre una posizione con i parametri specificati

Setting parameters	
PositionModify	Modifica i parametri della posizione con il simbolo o la posizione specificata del ticket
PositionClose	Chiude la posizione per il simbolo specificato
PositionClosePartial	Chiude parzialmente una posizione su un simbolo specificato in caso di un conto "hedging".
PositionCloseBy	Chiude la posizione con il ticket specificato da una posizione opposta
Metodi aggiuntivi	
Buy	Apre una posizione long con i parametri specificati
Sell	Apre una posizione short con i parametri specificati
BuyLimit	Piazza un ordine pendente di tipo Buy Limit con i parametri specificati
BuyStop	Piazza l'ordine pendente di tipo Buy Stop con i parametri specificati
SellLimit	Piazza l'ordine pendente di tipo Sell Limit con i parametri specificati
SellStop	Piazza l'ordine pendente di tipo Sell Stop con i parametri specificati
Accesso agli ultimi parametri richiesti	
Request	Ottiene la copia dell'ultima struttura request
RequestAction	Ottiene il tipo di operazione di trade
RequestActionDescription	Ottiene il tipo di operazione di trade come stringa
RequestMagic	Ottiene il numero magico dell' Expert Advisor
RequestOrder	Ottiene il ticket dell' ordine utilizzato nell' ultima richiesta
RequestSymbol	Ottiene il nome del simbolo utilizzato nell' ultima richiesta
RequestVolume	Ottiene il volume degli scambi (in lotti) utilizzato nell' ultima richiesta
RequestPrice	Ottiene il prezzo utilizzato nell ultima richiesta
RequestStopLimit	Ottiene il prezzo del'ordine pendente di tipo Stop Limit utilizzato nell' ultima richiesta
RequestSL	Ottiene il prezzo di Stop Loss dell'ordine utilizzato nell' ultima richiesta
RequestTP	Ottiene il prezzo Take Profit dell'ordine utilizzato nell' ultima richiesta

Setting parameters	
RequestDeviation	Ottiene la deviazione del prezzo dell'ordine utilizzato nell' ultima richiesta
RequestType	Ottiene il tipo di ordine utilizzato nell' ultima richiesta
RequestTypeDescription	Ottiene il tipo di ordine (come stringa) utilizzato nell' ultima richiesta
RequestTypeFilling	Ottiene il tipo di filling dell'ordine utilizzato nell' ultima richiesta
RequestTypeFillingDescription	Ottiene il tipo di filling dell'ordine (come stringa) utilizzato nell' ultima richiesta
RequestTypeTime	Ottiene il periodo di validità dell'ordine utilizzato nell' ultima richiesta
RequestTypeTimeDescription	Ottiene il periodo di validità dell'ordine (come stringa) utilizzato nell' ultima richiesta
RequestExpiration	Ottiene l'espirazione dell'ordine utilizzato nell' ultima richiesta
RequestComment	Ottiene il commento dell' ordine utilizzato nell' ultima richiesta
RequestPosition	Ottiene il ticket della posizione
RequestPositionBy	Ottiene il ticket della posizione opposta
L'accesso ai risultati di controllo dell'ultima richiesta	
CheckResult	Ottiene la copia dei risultati della struttura dell'ultima richiesta (request).
CheckResultRetcode	Ottiene il valore del campo retcode di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
CheckResultRetcodeDescription	Ottiene la stringa di descrizione del campo di retcode di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
CheckResultBalance	Ottiene il valore del campo di balance di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
CheckResultEquity	Ottiene il valore del campo di equity di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
CheckResultProfit	Ottiene il valore del campo profit di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
CheckResultMargin	Ottiene il valore del campo margin di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
CheckResultMarginFree	Ottiene il valore del campo margin_free di tipo MqlTradeCheckResult, riempito durante il controllo della

Setting parameters	
	correttezza della richiesta
CheckResultMarginLevel	Ottiene il valore del campo margin_level di tipoMqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
CheckResultComment	Ottiene il valore del campo comment di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta
L'accesso ai risultati di esecuzione della last request	
Result	Ottiene la copia dei risultato struttura last request
ResultRetcode	Ottiene il codice di risultato della richiesta
ResultRetcodeDescription	Ottiene il codice del risultato della richiesta come testo
ResultDeal	Ottiene il ticket dell'affare(deal)
ResultOrder	Ottiene il ticket dell'ordine
ResultVolume	Ottiene il volume dell'affare dell' ordine
ResultPrice	Ottiene il prezzo, confermato dal broker
ResultBid	Ottiene il corrente prezzo di bid (il requote)
ResultAsk	Ottiene il prezzo corrente (il requote) di ask
ResultComment	Ottiene il commento del broker
Metodi ausiliari	
PrintRequest	Stampa i parametri dell'ultima richiestanel journal
PrintResult	Stampa i risultati dell'ultima richiesta nel journal
FormatRequest	Prepara la stringa formattata con parametri dell'ultima richiesta
FormatRequestResult	Prepara la stringa formattata con risultati dell'esecuzione dell'ultima

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

LogLevel

Imposta livello di logging per i messaggi.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // livello di logging
   )
```

Parametri

log_level

[in] Livello di Logging.

Valore di ritorno

Nessuno.

Nota

LOG_LEVEL_NO e meno, disabilita la visualizzazione di ogni messaggio (impostati automaticamente in modalità di ottimizzazione). LOG_LEVEL_ERRORS permette la visualizzazione dei soli messaggi di errore (valore di default). LOG_LEVEL_ALL e più abilita la visualizzazione ogni messaggio (impostato automaticamente nella modalità di test).

ENUM_LOG_LEVELS

Identifier	Descrizione	Value
LOG_LEVEL_NO	Visualizzazione dei messaggi disabilitata	0
LOG_LEVEL_ERRORS	Vengono visualizzati solo i messaggi di errore	1
LOG_LEVEL_ALL	Vengono visualizzati tutti i messaggi	2

SetExpertMagicNumber

Imposta l'ID dell'expert.

```
void  SetExpertMagicNumber(
   ulong  magic      // ID
   )
```

Parametri

magic

[in] Nuovo ID dell'expert.

Valore di ritorno

Nessuno.

SetDeviationInPoints

Imposta la deviazione consentita.

```
void  SetDeviationInPoints(
   ulong  deviation      // deviazione
   )
```

Parametri

deviation

[in] Deviazione consentita.

Valore di ritorno

Nessuno.

SetTypeFilling

Imposta tipo di riempimento dell'ordine.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // tipo di riempimento dell'ordine
   )
```

Parametri

filling

[in] Tipo di riempimento dell'ordine (valore dell'enumerazione ENUM_ORDER_TYPE_FILLING).

Valore di ritorno

Nessuno.

SetTypeFillingBySymbol

Imposta il tipo di il riempimento (filling) dell'ordine in base alle impostazioni del simbolo specificato.

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // nome simbolo
   )
```

Parametri

symbol

[in] Nome del simbolo, in cui SYMBOL_FILLING_MODE contiene politiche di riempimento degli ordini consentiti

Valore di ritorno

true - esecuzione corretta, false - non si riusciti a definire la politica di riempimento.

Note

Se le politiche di riempimento SYMBOL_FILLING_FOK e SYMBOL_FILLING_IOC sono consentite per un simbolo, contemporaneamente, Il valore di ORDER_FILLING_FOK è impostato per l'ordine.

SetAsyncMode

Imposta la modalità asincrona per le operazioni di trade.

```
void  SetAsyncMode(
   bool  mode      // flag modalità asincrona
   )
```

Parametri

mode

[in] Flag modalità asincrona.

Valore di ritorno

Nessuno.

Nota

Questa modalità viene utilizzata per operazioni di trade asincrone (senza attendere la risposta del trade server ad una richiesta inviata) (vedi OrderSendAsync).

SetMarginMode

Imposta la modalità di calcolo del margine secondo le impostazioni dell'account.

```
void  SetMarginMode()
```

Valore di ritorno

No.

Nota

La modalità di calcolo del margine è specificata in ENUM_ACCOUNT_MARGIN_MODE.

OrderOpen

Piazza l'ordine pendnte con i parametri impostati.

```
bool  OrderOpen(
   const string          symbol,          // simbolo(symbol)
   ENUM_ORDER_TYPE       order_type,      // tipo di ordine
   double                volume,          // volume dell'ordine
   double                limit_price,     // prezzo StopLimit
   double                price,           // prezzo d'esecuzione
   double                sl,              // prezzo Stop Loss
   double                tp,              // prezzo Take Profit
   ENUM_ORDER_TYPE_TIME  type_time,       // tipo di espirazione
   datetime              expiration,      // espirazione
   const string          comment=""       // commento
   )
```

Parametri

symbol

[in] Nome dello strumento di trade.

order_type

[in] Tipo di operazione per il trade (valore dell'enumerazione ENUM_ORDER_TYPE).

volume

[in] Volume degli ordini richiesto.

limit_price

[in] prezzo al quale l'ordine StopLimit verrà piazzato.

price

[in] prezzo al quale l'ordine deve essere eseguito.

sl

[in] prezzo al quale il Stop Loss si innescherà.

tp

[in] prezzo al quale il Take Profit si innescherà.

type_time

[In] Tipo di ordine per esecuzione (valore dell'enumerazione ENUM_ORDER_TYPE_TIME).

expiration

[in] Data di scadenza dell' ordine pendente.

comment=""

[in] Commento dell'ordine.

Valore di ritorno

true - in caso di successo del controllo delle strutture di base, in caso contrario - false.

Nota

Il completamento con succeddo del metodo OrderSend(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (codice di ritorno del trade server) con ResultRetcode() ed il valore restituito da ResultOrder().

OrderModify

Modifica i parametri d'ordine pendente.

```
bool  OrderModify(
   ulong                ticket,         // ticket dell'ordine
   double               price,          // prezzo dell'esecuzione
   double               sl,             // prezzo Stop Loss
   double               tp,             // prezzo Take Profit
   ENUM_ORDER_TYPE_TIME type_time,      // tipo di espirazione
   datetime             expiration,     // espirazione
   double               stoplimit       // prezzo ordine Limit
   )
```

Parametri

ticket

[in] Ticket ordine.

price

[in] Il nuovo prezzo per cui l'ordine dev' essere eseguito (o il valore precedente, se il cambiamento non è necessario).

sl

[in] Il nuovo prezzo con il quale lo Stop Loss si attiverà (o il valore precedente, se il cambiamento non è necessario).

tp

[in] Il nuovo prezzo con il quale il Take Profit si attiverà (o il valore precedente, se il cambiamento non è necessario).

type_time

[in] Il nuovo tipo di ordine per espirazione (o il valore precedente, se il cambiamento non è necessario), valore dell'enumerazione ENUM_ORDER_TYPE_TIME.

expiration

[in] La nuova data di espirazione dell' ordine pendente (o il valore precedente, se il cambiamento non è necessario).

stoplimit

[in] Nuovo prezzo utilizzato per impostare un ordine Limit, quando il prezzo raggiunge il valore *prezzo* . Viene specificato solo per ordini StopLimit.

Valore di ritorno

true - in caso di successo del controllo delle strutture di base, in caso contrario - false.

Nota

Il completamento con successo del metodo OrderModify(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (codice di ritorno del trade server) con ResultRetcode().

OrderDelete

Elimina l'ordine pendente.

```
bool  OrderDelete(
   ulong  ticket      // ticket dell'ordine
   )
```

Parametri

ticket

[in] Ticket ordine.

Valore di ritorno

true - in caso di successo del controllo delle strutture di base, in caso contrario - false.

Nota

Il completamento con successo del metodo OrderDelete(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (codice di ritorno del trade server) con ResultRetcode().

PositionOpen

Apre una posizione con i parametri specificati.

```
bool  PositionOpen(
   const string     symbol,          // simbolo
   ENUM_ORDER_TYPE  order_type,      // tipo di ordine della posizione da aprire
   double           volume,          // volume della posizione
   double           price,           // prezzo d'esecuzione
   double           sl,              // prezzo Stop Loss
   double           tp,              // prezzo Take Profit
   const string     comment=""       // commento
   )
```

Parametri

symbol

[in] Nome strumento di trade, con cui si intende aprire la posizione.

order_type

[in] Tipo di ordine (operazione di trade) per aprire la posizione (valore dell'enumerazione ENUM_ORDER_TYPE).

volume

[in] Volume della posizione richiesta.

price

[in] Prezzo al quale la posizione dev' essere aperta.

sl

[in] prezzo al quale il Stop Loss si innescherà.

tp

[in] prezzo al quale il Take Profit si innescherà.

comment=""

[in] Commento della posizione.

Valore di ritorno

true - in caso di successo del controllo delle strutture di base, in caso contrario - false.

Nota

Il completamento di successo del metodo PositionOpen(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (codice di ritorno del trade server) con ResultRetcode() e il valore restituito da ResultDeal().

PositionModify

Modifica i parametri della posizione del simbolo specificato.

```
bool  PositionModify(
   const string  symbol,      // simbolo
   double        sl,          // prezzo Stop Loss
   double        tp           // prezzo Take Profit
   )
```

Modifica i parametri della posizione per ticket specificato.

```
bool  PositionModify(
   const ulong   ticket,      // ticket posizione
   double        sl,          // prezzo Stop Loss
   double        tp           // prezzo Take Profit
   )
```

Parametri

symbol

[in] Nome dello strumento di trade, con il quale si intende modificare la posizione.

ticket

[in] biglietteria della posizione da modificare.

sl

[in] Il nuovo prezzo con il quale lo Stop Loss si attiverà (o il valore precedente, se il cambiamento non è necessario).

tp

[in] Il nuovo prezzo con il quale il Take Profit si attiverà (o il valore precedente, se il cambiamento non è necessario).

Valore di ritorno

true - in caso di successo del controllo delle strutture di base, in caso contrario - false.

Nota

Il completamento di successo del metodo PositionModify(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (codice di ritorno del trade server) con ResultRetcode().

Per l'interpretazione "netting" delle posizioni (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE), solo una posizione può esistere per un simbolo in qualsiasi momento di tempo. Questa posizione è il risultato di uno o più affari. Non confondere con posizioni valide con ordini pendenti, che vengono visualizzati anche nella scheda Trading della finestra Attrezzi.

Se singole posizioni sono ammesse (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), più posizioni possono essere aperte per un simbolo. In questo caso, PositionModify modificherà una posizione con il ticket più basso.

PositionClose

Chiude una posizione dal simbolo specificato.

```
bool  PositionClose(
   const string  symbol,                  // simbolo
   ulong         deviation=ULONG_MAX      // deviazione
   )
```

Si chiude una posizione con il ticket specificato.

```
bool  PositionClose(
   const ulong   ticket,                  // Ticket della posizione
   ulong         deviation=ULONG_MAX      // Deviazione
   )
```

Parametri

symbol

[in] Nome dello strumento di trade, mediante il quale si intende chiudere la posizione.

ticket

[in] Ticket di una posizione chiusa.

deviation=ULONG_MAX

[in] Deviazione massima dal prezzo corrente (in punti).

Valore di ritorno

true - in caso di successo del controllo delle strutture di base, in caso contrario - false.

Nota

Il completamento cn successo del metodo PositionClose(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (codice di ritorno del trade server) con ResultRetcode().

Per l'interpretazione "netting" delle posizioni (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE), solo una posizione può esistere per un simbolo in qualsiasi momento di tempo. Questa posizione è il risultato di uno o più affari. Non confondere con posizioni valide con ordini pendenti, che vengono visualizzati anche nella scheda Trading della finestra Attrezzi.

Se singole posizioni sono ammesse (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), più posizioni possono essere aperte per un simbolo. In questo caso, PositionClose chiuderà una posizione con il ticket più basso.

PositionClosePartial

Chiude parzialmente una posizione su un simbolo specificato in caso di un conto "hedging".

```
bool  PositionClosePartial(
   const string  symbol,                  // simbolo
   const double  volume,                  // volume
   ulong         deviation=ULONG_MAX      // deviazione
   )
```

Chiude parzialmente una posizione avendo un ticket specificato in caso di un conto "hedging".

```
bool  PositionClosePartial(
   const ulong   ticket,                  // ticket della posizione
   const double  volume,                  // volume
   ulong         deviation=ULONG_MAX      // deviazione
   )
```

Parametri

symbol

[in] Nome di uno strumento di trading sulla quale una posizione è parzialmente chiusa. Se viene specificato un simbolo (non un ticket) per una chiusura parziale della posizione, viene selezionata la prima posizione rilevata con il MagicNumber specificato (Expert Advisor ID) sul simbolo. Perciò, a volte è meglio usare PositionClosePartial() con il ticket della posizione specificato.

volume

[in] Volume, con la quale una posizione dovrebbe essere ridotta. Se il valore eccede il volume di una posizione parzialmente chiusa, viene chiusa per intero. Nessuna posizione nella direzione opposta viene aperta.

ticket

[in] Ticket della posizione chiusa.

deviation=ULONG_MAX

[in] La deviazione massima dal prezzo corrente (in punti).

Valore Restituito

true se il controllo di base delle strutture ha successo, altrimenti false.

Nota

La riuscita con successo del metodo PositionClosePartial(...) non significa sempre un'esecuzione riuscita di un'operazione di trading. Si dovrebbe chiamare il metodo ResultRetcode() per controllare il risultato di una richiesta di trade (codice di ritorno del server di trade).

Nel sistema "netting" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE), per ogni simbolo, in un dato momento solo una posizione può essere aperta, che è il risultato di uno o più deals. Non confondere gli attuali ordini pendenti con le posizioni che vengono anche visualizzate nella scheda "Trade" del pannello "BoxAttrezzi" del terminale client.

In caso di rappresentazione della posizione (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), è possibile aprire simultaneamente più posizioni su ciascun simbolo. In questo caso, PositionClose chiude una posizione che ha almeno un ticket.

PositionCloseBy

Si chiude una posizione con il ticket specificato da una posizione opposta.

```
bool  PositionCloseBy(
   const ulong   ticket,        // Ticket posizione
   const ulong   ticket_by      // Ticket posizione opposta
   )
```

Parametri

ticket

[in] Ticket della posizione chiusa.

ticket_by

[in] Ticket della posizione opposta utilizzato per la chiusura.

Valore restituito

true se il controllo di base delle strutture ha successo, altrimenti false.

Nota

Il completamento con successo del metodo PositionCloseBy(...) non significa sempre una corretta esecuzione di un'operazione di trading. Si dovrebbe chiamare il metodo ResultRetcode() per controllare il risultato della richiesta di trade (codice di ritorno del trade server).

Buy

Apre una posizione long con i parametri specificati.

```
bool  Buy(
   double         volume,          // volume della posizione
   const string   symbol=NULL,     // simbolo
   double         price=0.0,       // prezzo
   double         sl=0.0,          // prezzo stop loss
   double         tp=0.0,          // prezzo take profit
   const string   comment=""       // commento
   )
```

Parametri

volume

[in] Volume della posizione.

symbol=NULL

[in] Simbolo della posizione. Se il simbolo non è specificato, verrà utilizzato il simbolo corrente.

price=0.0

[in] Prezzo. Se il prezzo non è specificato, verrà usato il corrente prezzo Ask del mercato.

sl=0.0

[in] Prezzo Stop Loss.

tp=0.0

[in] Prezzo Take Profit.

comment=""

[in] Commento.

Valore di ritorno

true - in caso di successo del controllo della struttura, altrimenti false.

Nota

Il completamento di successo del metodo Buy (...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (trade server codice di ritorno) usando ResultRetcode() e il valore restituito da ResultDeal().

Sell

Apre una posizione short con i parametri specificati.

```
bool  Sell(
   double         volume,          // volume della posizione
   const string   symbol=NULL,     // simbolo
   double         price=0.0,       // prezzo
   double         sl=0.0,          // prezzo stop loss
   double         tp=0.0,          // prezzo take profit
   const string   comment=""       // commento
   )
```

Parametri

volume

[in] Volume della posizione.

symbol=NULL

[in] Simbolo della posizione. Se il simbolo non è specificato, verrà utilizzato il simbolo corrente.

price=0.0

[in] Prezzo. Se il prezzo non è specificato, verrà utilizzato il corrente prezzo Bid di mercato.

sl=0.0

[in] Prezzo Stop Loss.

tp=0.0

[in] Prezzo Take Profit.

comment=""

[in] Commento.

Valore di ritorno

true - in caso di successo del controllo della struttura, altrimenti false.

Nota

Il completamento di successo del metodo Sell(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (trade server codice di ritorno) usando ResultRetcode() e il valore restituito da ResultDeal().

BuyLimit

Piazza l'ordine pendente di tipo Buy Limit (comprare a prezzo inferiore al prezzo corrente di mercato) con i parametri specificati.

```
bool  BuyLimit(
   double                volume,                    // volume ordine
   double                price,                     // prezzo ordine
   const string          symbol=NULL,               // simbolo
   double                sl=0.0,                    // prezzo stop loss
   double                tp=0.0,                    // prezzo take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // lifetime dell'ordine
   datetime              expiration=0,              // orario di espirazione dell'
   const string          comment=""                 // commento
   )
```

Parametri

volume

[in] Volume dell'ordine.

price

[in] Prezzo dell'ordine.

symbol=NULL

[in] Simbolo dell'ordine. Se il simbolo non è specificato, verrà utilizzato il simbolo corrente.

sl=0.0

[in] Prezzo Stop Loss.

tp=0.0

[in] Prezzo Take Profit.

type_time=ORDER_TIME_GTC

[in] Lifetime dell'ordine (valore dell'enumerazione ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Orario di espirazione(scadenza) dell'ordine (usato solo se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commento dell'ordine.

Valore di ritorno

true - in caso di successo del controllo della struttura, altrimenti false.

Nota

Il completamento con successo del metodo BuyLimit(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (trade server codice di ritorno) usando ResultRetcode() e il valore restituito da ResultOrder().

BuyStop

Piazza l'ordine pendente di tipo Buy Stop (comprare a prezzo superiore al prezzo corrente di mercato) con i parametri specificati.

```
bool  BuyStop(
   double                volume,                    // volume ordine
   double                price,                     // prezzo ordine
   const string          symbol=NULL,               // simbolo
   double                sl=0.0,                    // prezzo stop loss
   double                tp=0.0,                    // prezzo take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // lifetime dell'ordine
   datetime              expiration=0,              // orario di espirazione dell'
   const string          comment=""                 // commento
   )
```

Parametri

volume

[in] Volume dell'ordine.

price

[in] Prezzo dell'ordine.

symbol=NULL

[in] Simbolo dell'ordine. Se il simbolo non è specificato, verrà utilizzato il simbolo corrente.

sl=0.0

[in] Prezzo Stop Loss.

tp=0.0

[in] Prezzo Take Profit.

type_time=ORDER_TIME_GTC

[in] Lifetime dell'ordine (valore dell'enumerazione ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Orario d'espirazione dell'ordine (usato solo se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commento dell'ordine.

Valore di ritorno

true - in caso di successo del controllo della struttura, altrimenti false.

Nota

Il completamento di successo del metodo BuyStop(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (trade server codice di ritorno) usando ResultRetcode() e il valore restituito da ResultOrder().

SellLimit

Piazza l'ordine pendente di tipo Sell Limit (vende al prezzo più alto rispetto al prezzo corrente di mercato) con parametri specificati.

```
bool  SellLimit(
   double                volume,                    // volume ordine
   double                price,                     // prezzo ordine
   const string          symbol=NULL,               // simbolo
   double                sl=0.0,                    // prezzo stop loss
   double                tp=0.0,                    // prezzo take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // lifetime dell'ordine
   datetime              expiration=0,              // orario di espirazione dell'
   const string          comment=""                 // commento
   )
```

Parametri

volume

[in] Volume dell'ordine.

price

[in] Prezzo dell'ordine.

symbol=NULL

[in] Simbolo dell'ordine. Se il simbolo non è specificato, verrà utilizzato il simbolo corrente.

sl=0.0

[in] Prezzo Stop Loss.

tp=0.0

[in] Prezzo Take Profit.

type_time=ORDER_TIME_GTC

[in] Lifetime dell'ordine (valore dell'enumerazione ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Orario d'espirazione dell'ordine (usato solo se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commento dell'ordine.

Valore di ritorno

true - in caso di successo del controllo della struttura, altrimenti false.

Nota

Il completamento di successo del metodo SellLimit(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (trade server codice di ritorno) usando ResultRetcode() e il valore restituito da ResultOrder().

SellStop

Piazza l'ordine in attesa di tipo Sell Stop (vende al prezzo inferiore al prezzo corrente di mercato) con i parametri specificati.

```
bool  SellStop(
   double                volume,                    // volume ordine
   double                price,                     // prezzo ordine
   const string          symbol=NULL,               // simbolo
   double                sl=0.0,                    // prezzo stop loss
   double                tp=0.0,                    // prezzo take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // lifetime dell'ordine
   datetime              expiration=0,              // orario di espirazione dell'
   const string          comment=""                 // commento
   )
```

Parametri

volume

[in] Volume dell'ordine.

price

[in] Prezzo dell'ordine.

symbol=NULL

[in] Simbolo dell'ordine. Se il simbolo non è specificato, verrà utilizzato il simbolo corrente.

sl=0.0

[in] Prezzo Stop Loss.

tp=0.0

[in] Prezzo Take Profit.

type_time=ORDER_TIME_GTC

[in] Lifetime dell'ordine (valore dell'enumerazione ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Orario d'espirazione dell'ordine (usato solo se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commento dell'ordine.

Valore di ritorno

true - in caso di successo del controllo della struttura, altrimenti false.

Nota

Il completamento di successo del metodo SellStop(...) non sempre significa esecuzione riuscita dell'operazione di trade. E' necessario verificare il risultato della richiesta di trade (trade server codice di ritorno) usando ResultRetcode() e il valore restituito da ResultOrder().

Request

Ottiene la copia dell'ultima struttura richiesta.

```
void  Request(
   MqlTradeRequest&  request      // struttura target
   ) const
```

Parametri

request

[out] Riferimento alla struttura di tipo MqlTradeRequest.

Valore di ritorno

Nessuno.

RequestAction

Ottiene il tipo di operazione di trade.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

Valore di ritorno

Tipo d'operazione di trade usata nell'ultima richiesta.

RequestActionDescription

Ottiene il tipo di operazione di trade come stringa.

```
string  RequestActionDescription() const
```

Valore di ritorno

Tipo di operazione di trade (come stringa) utilizzata nell'ultima richiesta.

RequestMagic

Ottiene il numero magico dell' Expert Advisor.

```
ulong  RequestMagic() const
```

Valore di ritorno

Il numero magico (ID) dell' Expert Advisor, utilizzato nell'ultima richiesta.

RequestOrder

Ottiene il ticket dell'ordine utilizzato nella ultima richiesta.

```
ulong  RequestOrder() const
```

Valore di ritorno

Ticket dell'ordine dell' ultima richiesta.

RequestSymbol

Ottiene il nome del simbolo utilizzato nell'ultima richiesta.

```
string  RequestSymbol() const
```

Valore di ritorno

Il nome del simbolo utilizzato nell'ultima richiesta.

RequestVolume

Ottiene il volume degli scambi (in lotti) utilizzato nella ultima richiesta.

```
double  RequestVolume() const
```

Valore di ritorno

Il volume degli scambi (in lotti) utilizzato nella ultima richiesta.

RequestPrice

Ottiene il prezzo utilizzato nell'ultima richiesta.

```
double  RequestPrice() const
```

Valore di ritorno

Prezzo dell'ordine utilizzato nella ultima richiesta.

RequestStopLimit

Ottiene il prezzo dell'ordine pendente di tipo Stop Limit utilizzato nell' ultima richiesta.

```
double  RequestStoplimit() const
```

Valore di ritorno

Il prezzo dell'ordine pendente di tipo Stop Limit utilizzato nell' ultima richiesta.

RequestSL

Ottiene il prezzo Stop Loss dell'ordine utilizzato nell' ultima richiesta.

```
double  RequestSL() const
```

Valore di ritorno

Il prezzo di Stop Loss utilizzato nell'ultima richiesta.

RequestTP

Ottiene il prezzo Take Profit dell'ordine utilizzato nella ultima richiesta.

```
double  RequestTP() const
```

Valore di ritorno

Il prezzo Take Profit utilizzato nel l'ultima richiesta.

RequestDeviation

Ottiene la deviazione prezzo dell'ordine utilizzato nella ultima richiesta.

```
ulong  RequestDeviation() const
```

Valore di ritorno

La deviazione prezzo dell'ordine utilizzato nell'ultima richiesta.

RequestType

Ottiene il tipo di ordine usato nell' ultima richiesta.

```
ENUM_ORDER_TYPE  RequestType() const
```

Valore di ritorno

Tipo di ordine usato nella ultima richiesta (valore dell'enumerazione ENUM_ORDER_TYPE).

RequestTypeDescription

Ottiene il tipo di ordine (come stringa) utilizzato nell' ultima richiesta.

```
string  RequestTypeDescription() const
```

Valore di ritorno

Il tipo di ordine (come stringa) utilizzato nell' ultima richiesta.

RequestTypeFilling

Ottiene il tipo di riempimento dell'ordine utilizzato nell'ultima richiesta.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

Valore di ritorno

Il tipo di riempimento dell'ordine (valore di ENUM_ORDER_TYPE_FILLING) utilizzato nell' ultima richiesta.

RequestTypeFillingDescription

Ottiene il tipo di riempimento dell'ordine (come stringa) utilizzato nell' ultima richiesta.

```
string  RequestTypeFillingDescription() const
```

Valore di ritorno

Il tipo di riempimento (come stringa) dell'ordine utilizzato nell'ultima richiesta.

RequestTypeTime

Ottiene il periodo di validità dell'ordine utilizzato nell'ultima richiesta.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

Valore di ritorno

Il periodo di validità dell'ordine (valore dell'enumerazione ENUM_ORDER_TYPE_TIME) utilizzato nell'ultima richiesta.

RequestTypeTimeDescription

Ottiene il periodo di validità dell'ordine (come stringa) utilizzato nell'ultima richiesta.

```
string  RequestTypeTimeDescription() const
```

Valore di ritorno

Il periodo di validità dell'ordine (come stringa) utilizzato nell'ultima richiesta.

RequestExpiration

Ottiene il tempo di scadenza dell'ordine utilizzato nella ultima richiesta.

```
datetime  RequestExpiration() const
```

Valore di ritorno

Il tempo di scadenza dell'ordine utilizzato nell'ultima richiesta.

RequestComment

Ottiene il commento dell 'ordine utilizzato nella ultima richiesta.

```
string  RequestComment() const
```

Valore di ritorno

Il commento dell' ordine utilizzato nella ultima richiesta.

RequestPosition

Ottiene il ticket della posizione.

```
ulong  RequestPosition() const
```

Return value

Ticket della posizione utilizzata nell'ultima richiesta.

RequestPositionBy

Ottiene il ticket della posizione opposta.

```
ulong  RequestPositionBy() const
```

Return value

Ticket della posizione utilizzato nell' ultima richiesta.

Result

Ottiene la copia della struttura del risultato dell' ultima richiesta.

```
void  Result(
   MqlTradeResult&  result      // riferimento
   ) const
```

Parametri

result

[out] Riferimento alla struttura di tipo MqlTradeResult.

Valore di ritorno

Nessuno.

ResultRetcode

Ottiene il codice di richiesta del risultato.

```
uint  ResultRetcode() const
```

Valore di ritorno

Il Codice del risultato della richiesta.

ResultRetcodeDescription

Ottiene il codice del risultato della richiesta come testo.

```
string  ResultRetcodeDescription() const
```

Valore di ritorno

Codice del risultato dell'ultima richiestacome testo.

ResultDeal

Ottiene il ticket.

```
ulong  ResultDeal() const
```

Valore di ritorno

Ticket dell'affare, se si esegue l'operazione.

ResultOrder

Ottiene il ticket dell'ordine.

```
ulong  ResultOrder() const
```

Valore di ritorno

Ticket dell'ordine, se l'ordine è piazzato.

ResultVolume

Ottiene il volume dell' affare o dell' ordine.

```
double  ResultVolume() const
```

Valore di ritorno

Volume dell' affare o dell'ordine.

ResultPrice

Ottiene il prezzo, confermato dal broker.

```
double  ResultPrice() const
```

Valore di ritorno

Prezzo, confermato dal broker.

ResultBid

Ottiene il prezzo Bid corrente (la riquotazione).

```
double  ResultBid() const
```

Valore di ritorno

Corrente prezzo bid (il requote).

ResultAsk

Ottiene il prezzo Ask corrente (la riquotazione).

```
double  ResultAsk() const
```

Valore di ritorno

Corrente prezzo ask (il requote).

ResultComment

Ottiene il commento broker.

```
string  ResultComment() const
```

Valore di ritorno

Commento broker all'operazione.

CheckResult

Ottiene la copia dei risultati della struttura dell'ultima richiesta (request).

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // riferimento
   ) const
```

Parametri

check_result

[out] Riferimento alla struttura target di tipo MqlTradeCheckResult.

Valore di ritorno

Nessuno.

CheckResultRetcode

Ottiene il valore del campo retcode di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta

```
uint  CheckResultRetcode() const
```

Valore di ritorno

Il valore del campo retcode (codice errore) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultRetcodeDescription

Ottiene la stringa di descrizione del campo di retcode di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta

```
string  ResultRetcodeDescription() const
```

Valore di ritorno

La stringa di descrizione del campo retcode (codice di errore) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultBalance

Ottiene il valore del campo balance di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

```
double  CheckResultBalance() const
```

Valore di ritorno

Il valore del campo balance (valore del bilancio che sarà, dopo l'esecuzione dell'operazione di trade) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultEquity

Ottiene il valore del campo di equity di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta

```
double  CheckResultEquity() const
```

Valore di ritorno

Il valore del campo balance (valore dell'equità(quity) che sarà dopo l'esecuzione dell'operazione di trade) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultProfit

Ottiene il valore del campo profit di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta

```
double  CheckResultProfit() const
```

Valore di ritorno

Il valore del campo profit (valore del profitto che sarà dopo l'esecuzione dell'operazione di trade) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultMargin

Ottiene il valore del campo margin di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta

```
double  CheckResultMargin() const
```

Valore di ritorno

Il valore del campo margin (margine richiesto per l'operazione di trade) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultMarginFree

Ottiene il valore del campo margin_free di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta

```
double  CheckResultMarginFree() const
```

Valore di ritorno

Il valore del campo margin_free (margine libero che verrà lasciato dopo l'esecuzione dell'operazione di trade) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultMarginLevel

Ottiene il valore del campo margin_level di tipoMqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta

```
double  CheckResultMarginLevel() const
```

Valore di ritorno

Il valore del campo margin_level (livello del margine che verrà impostato dopo l'esecuzione dell'operazione di trade) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

CheckResultComment

Il valore del campo commento di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

```
string  CheckResultComment() const
```

Valore di ritorno

Il valore del campo commento (Commento al codice di risposta, descrizione errore) di tipo MqlTradeCheckResult, riempito durante il controllo della correttezza della richiesta.

PrintRequest

Consente di stampare gli ultimi parametri della richiesta nel journal.

```
void  PrintRequest() const
```

Valore di ritorno

Nessuno.

PrintResult

Stampa i risultati dell'ultima richiesta nel journal.

```
void  PrintResult() const
```

Valore di ritorno

Nessuno.

FormatRequest

Prepara la stringa formattata con i parametri dell' ultima richiesta.

```
string  FormatRequest(
   string&                 str,          // stringa target
   const MqlTradeRequest&  request       // richiesta
   ) const
```

Parametri

str

[in] Stringa di destinazione(target), passato per riferimento.

request

[in] La struttura di tipo MqlTradeRequestcon i parametri dell'ultima richiesta.

Valore di ritorno

Nessuno.

FormatRequestResult

Prepara la stringa formattata con risultati dell'esecuzione dell'ultima richiesta.

```
string  FormatRequestResult(
   string&                 str,          // stringa
   const MqlTradeRequest&  request,      // struttura request
   const MqlTradeResult&   result        // struttura result
   ) const
```

Parametri

str

[in] Stringa di destinazione(target), passato per riferimento.

request

[in] La struttura di tipo MqlTradeRequest con parametri della ultima richiesta.

result

[in] La struttura di tipo MqlTradeResult con risultati dell'ultima richiesta.

Valore di ritorno

Nessuno.

CTerminalInfo

CTerminalInfo è una classe per l'accesso semplificato alle proprietà di ambiente programmazione mql5.

Descrizione

La Classe CTerminalInfo fornisce l'accesso alle proprietà di ambiente programmazione mql5.

Dichiarazione

```
class CTerminalInfo : public CObject
```

Titolo

```
#include <Trade\TerminalInfo.mqh>
```

Gerarchia di ereditarietà

CObject

CTerminalInfo

I metodi di classe per gruppi

Metodi per l'accesso alle proprietà di tipo integer	
Build	Ottiene il numero di build del terminale client
IsConnected	Ottiene le informazioni sulla connessione al trade server
IsDLLsAllowed	Ottiene le informazioni sul permesso di utilizzo di DLL
IsTradeAllowed	Ottiene le informazioni relative alle autorizzazioni al trade
IsEmailEnabled	Ottiene le informazioni sul permesso di inviare e-mail ai server SMTP ed il login, specificato nelle impostazioni del terminale
IsFtpEnabled	Ottiene le informazioni sul permesso di inviare i reports di trade al server FTP ed effettuare il login, specificato nelle impostazioni del terminale
MaxBars	Ottiene le informazioni sul numero massimo di barre sul grafico
CodePage	Ottiene le informazioni sulla pagina di codice della lingua nel terminale client
CPUCores	Ottiene le informazioni sui core della CPU
MemoryPhysical	Ottiene le informazioni sulla memoria fisica (in Mb)
MemoryTotal	Ottiene le informazioni sulla memoria totale, disponibile per il processo del terminale/agente (in Mb)
MemoryAvailable	Ottiene le informazioni sulla memoria libera, disponibile per il processo del terminali/agente (in Mb)

Metodi per l'accesso alle proprietà di tipo integer	
MemoryUsed	Ottiene le informazioni sulla memoria, utilizzata dal processo terminale/agente (in Mb)
IsX64	Ottiene le informazioni relative al tipo di terminale cliente (32/64 bit)
OpenCLSupport	Ottiene le informazioni sulla versione di OpenCL, supportata da scheda video
DiskSpace	Ottiene le informazioni sullo spazio libero su disco (in MB)
Metodi per l'accesso alle proprietà di tipo stringa	
Language	Ottiene la lingua del terminale client
Name	Ottiene il nome del terminale client
Company	Ottiene il nome della società del terminale client
Path	Ottiene la cartella del terminale client
DataPath	Ottiene la cartella dei dati del terminale client
CommonDataPath	Ottiene la cartella dati comune di tutti i terminali client, installati sul computer
L'accesso alle funzioni API MQL5	
InfoInteger	Ottiene il valore della proprietà di tipo integer
InfoString	Ottiene il valore della proprietà di tipo string

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Build

Ottiene il numero di build del terminale client.

```
int  CBuild() const
```

Valore di ritorno

Numero di build del terminale client.

Nota

Per ottenere il numero di build utilizza la funzione TerminalInfoInteger()(proprietà TERMINAL_BUILD).

IsConnected

Ottiene le informazioni sulla connessione al trade server.

```
bool  IsConnected() const
```

Valore di ritorno

true, se il terminale è collegato al trade server, altrimenti false.

Nota

Per ottenere lo stato della connessione utilizza la funzione TerminalInfoInteger() (propeirtà TERMINAL_CONNECTED).

IsDLLsAllowed

Ottiene le informazioni sul permesso di utilizzo DLL.

```
bool  IsDLLsAllowed() const
```

Valore di ritorno

true, se è consentito l'utilizzo DLL, altrimenti false.

Nota

Per ottenere informazioni su permessi di utilizzo DLL utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_DLLS_ALLOWED).

IsTradeAllowed

Ottiene le informazioni relative al permesso di trade.

```
bool  IsTradeAllowed() const
```

Valore di ritorno

vero, se il trade è consentito, altrimenti false.

Nota

Per avere informazioni sulle autorizzazioni per il trading utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_TRADE_ALLOWED).

IsEmailEnabled

Ottiene le informazioni sul permesso di inviare e-mail ai server SMTP ed il login, specificato nelle impostazioni del terminale.

```
bool  IsEmailEnabled() const
```

Valore di ritorno

true, se è consentito l'invio di e-mail, altrimenti false.

Nota

Per avere informazioni sul permesso di invio e-mail utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_EMAIL_ENABLED).

IsFtpEnabled

Ottiene le informazioni sul permesso di inviare i reports di trade al server FTP ed effettuare il login, specificato nelle impostazioni del terminale.

```
bool  IsFtpEnabled() const
```

Valore di ritorno

true, se l'invio trade reports al server FTP è consentito, altrimenti false.

Nota

Per ottenere informazioni su permesso di inviare i trade reportsutilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_FTP_ENABLED).

MaxBars

Ottiene il numero massimo di barre sul grafico, specificate nelle impostazioni del terminale client.

```
int  MaxBars() const
```

Valore di ritorno

Numero massimo di barre sul chart.

Nota

Per ottenere il massimo numero di barre sul chart utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_MAXBARS).

CodePage

Ottiene le informazioni sulla pagina di codice della lingua nel terminale client.

```
int  CodePage() const
```

Valore di ritorno

Pagina Codice della lingua nel terminale client.

Nota

Per ottenere la pagina di codice utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_CODEPAGE).

CPUCores

Ottiene le informazioni sulla quantità di core della CPU del sistema.

```
int  CPUCores() const
```

Valore di ritorno

Quantità di core di CPU nel sistema.

Nota

Per ottenere la quantità di core della CPU utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_CPU_CORES).

MemoryPhysical

Ottiene le informazioni sulla memoria fisica (in Mb).

```
int  MemoryPhysical() const
```

Valore di ritorno

Memoria fisica (in Mb).

Nota

Per ottenere la memoria fisica utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_MEMORY_PHYSICAL).

MemoryTotal

Ottiene le informazioni sulla memoria totale, disponibile per il terminale/agente client (in Mb).

```
int  MemoryTotal() const
```

Valore di ritorno

Memoria totale (in MB), disponibile per il terminale/agente.

Nota

Per ottenere la memoria totale utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_MEMORY_TOTAL).

MemoryAvailable

Ottiene le informazioni sulla memoria libera, disponibile per il terminale/agente client (in Mb).

```
int  MemoryTotal() const
```

Valore di ritorno

Memoria libera (in Mb), disponibile per il terminale/agente.

Nota

Per ottenere la memoria libera utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_MEMORY_TOTAL).

MemoryUsed

Ottiene le informazioni sulla memoria, utilizzato dal terminale/agente client (in Mb).

```
int  MemoryUsed() const
```

Valore di ritorno

La memoria, utilizzata dal terminale/agente client (in Mb).

Nota

Per ottenere la memoria, utilizzata dal terminale utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_MEMORY_USED).

IsX64

Ottiene le informazioni sul tipo di terminale client.

```
bool  IsX64() const
```

Valore di ritorno

true, se si utilizza la versione a 64 bit, altrimenti false.

Nota

Per ottenere il tipo di terminale client utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_X64).

OpenCLSupport

Ottiene le informazioni sulla versione di OpenCL, supportatalla scheda video.

```
int  OpenCLSupport() const
```

Valore di ritorno

Il valore restituito ha la seguente forma: 0x00010002 = "1.2". Lo 0 significa che OpenCL non è supportato.

Nota

Per ottenere la versione di OpenCL utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_OPENCL_SUPPORT).

DiskSpace

Ottiene le informazioni sullo spazio libero su disco, disponibile per il terminale/agente client (in Mb).

```
int  MDiskSpace() const
```

Valore di ritorno

Spazio libero su disco (in MB), disponibile per il terminale/agente client (per i file, salvati nella cartella MQL5\File).

Nota

Per ottenere lo spazio libero utilizza la funzione TerminalInfoInteger() (proprietà TERMINAL_DISK_SPACE).

Language

Ottiene l'informazione sulla lingua nel terminale client.

```
string  Language() const
```

Valore di ritorno

Lingua, utilizzata nel terminale client.

Nota

Per ottenere la lingua usa la funzione TerminalInfoString() (proprietà TERMINAL_LANGUAGE).

Name

Ottiene le informazioni del nome del terminale client.

```
string  Name() const
```

Valore di ritorno

Nome del terminale client.

Nota

Per ottenere il nome del terminale client utilizza la funzione TerminalInfoString() (proprietà TERMINAL_NAME).

Company

Ottiene le informazioni relative al nome del broker.

```
string  Company() const
```

Valore di ritorno

Il nome del broker.

Nota

Per ottenere il nome mediatore utilizza la funzione TerminalInfoString() (proprietà TERMINAL_COMPANY).

Path

Ottiene la cartella del terminale client.

```
string  Path() const
```

Valore di ritorno

La cartella del terminale client.

Nota

Per ottenere la cartella terminale client utilizza la funzione TerminalInfoString() (proprietà TERMINAL_PATH).

DataPath

Ottiene il informazioni sulla cartella dei dati del terminale.

```
string  DataPath() const
```

Valore di ritorno

Cartella dei dati del terminale client.

Nota

Per ottenere la cartella dei dati del terminale client utilizza la funzione TerminalInfoString() (proprietà TERMINAL_DATA_PATH).

CommonDataPath

Ottiene la cartella dati comune di tutti i terminali client, installati sul computer.

```
string  CommonDataPath() const
```

Valore di ritorno

Cartella dei dati comuni.

Nota

Per ottenere cartella dei dati comuni utilizza la funzione TerminalInfoString() (propeirtà COMMON_DATA_PATH).

InfoInteger

Restituisce il valore di una proprietà corrispondente dell'ambiente programma di MQL5.

```
int  TerminalInfoInteger(
   int  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Può essere uno dei valori dell'enumerazione ENUM_TERMINAL_INFO_INTEGER.

Valore di ritorno

Valore di tipo int.

Nota

Per ottenere il valore della proprietà utilizza la funzione TerminalInfoInteger().

InfoString

La funzione restituisce il valore di una proprietà corrispondente dell'ambiente programma di MQL5. La proprietà deve essere di tipo string.

```
string  TerminalInfoString(
   int  property_id      // identificatore della proprietà
   );
```

Parametri

property_id

[in] Identificatore della proprietà. Forse uno dei valori dell'enumerazione ENUM_TERMINAL_INFO_STRING.

Valore di ritorno

Valore di tipo string.

Nota

Per ottenere il valore della proprietà utilizza la funzione TerminalInfoString().

Classi di Strategia di Trading

Questa sezione contiene i dettagli tecnici di lavoro con le classi per la creazione e la sperimentazione di strategie di trading e descrizione dei componenti rilevanti della libreria standard MQL5.

L'uso di queste classi vi farà risparmiare tempo durante la creazione di strategie di trading.

La MQL5 standard Library (in termini di strategie di trading) viene inserita nella directory del terminale, nella cartella Include\Expert.

Classi Base	Descrizione
CExpertBase	Classe base per tutte le strategie di trading
CExpert	Classe base per l'Expert Advisor
CExpertSignal	Classe base per le classi Trading Signal (Segnali di Trading)
CExpertTrailing	Classe base per le classi Trailing Stop
CExpertMoney	Classe base per le classi di Money Management (Gestione del denaro)

Classi dei segnali di Trading	Descrizione
CSignalAC	Il modulo di segnali basati su modelli di mercato dell'indicatore Accelerator Oscillator.
CSignalAMA	Il modulo di segnali basati su modelli di mercato dell'indicatore Adaptive Moving Average.
CSignalAO	Il modulo di segnali basati su modelli di mercato dell'indicatore Awesome Oscillator.
CSignalBearsPower	Il modulo di segnali basati su modelli di mercato dell'indicatore Bears Power.
CSignalBullsPower	Il modulo di segnali basati su modelli di mercato dell'indicatore Bulls Power.
CSignalCCI	Il modulo di segnali basati su modelli di mercato dell'indicatore Commodity Channel Index.
CSignalDeM	Il modulo di segnali basati su modelli di mercato dell'indicatore DeMarker.
CSignalDEMA	Il modulo di segnali sulla base di modelli di mercato dell'indicatore Double Exponential Moving Average.
CSignalEnvelopes	Il modulo di segnali basati su modelli di mercato dell'indicatore Envelopes.
CSignalFrAMA	Il modulo di segnali basati su modelli di mercato dell'indicatore Fractal Adaptive Moving Average.

Classi dei segnali di Trading	Descrizione
CSignalITF	Il modulo di filtrazione dei segnali per orario
CSignalMACD	Il modulo di segnali basati su modelli di mercato dell'indicatore MACD.
CSignalMA	Il modulo di segnali basati su modelli di mercato dell'indicatore Moving Average.
CSignalSAR	Il modulo di segnali basati su modelli di mercato dell'indicatore Parabolic SAR.
CSignalRSI	Il modulo di segnali basati su modelli di mercato dell'indicatore Relative Strength Index
CSignalRVI	Il modulo di segnali basati su modelli di mercato dell'indicatore Relative Vigor Index.
CSignalStoch	Il modulo di segnali basati su modelli di mercato dell'indicatore Stochastic.
CSignalTRIX	Il modulo di segnali sulla base di modelli di mercato dell'indicatore Triple Exponential Average..
CSignalTEMA	Il modulo di segnali sulla base di modelli di mercato dell'indicatore Triple Exponential Moving Average.
CSignalWPR	Il modulo di segnali basati su modelli di mercato dell'indicatore Williams Percent Range.

Trailing Stop classes	Descrizione
CTrailingFixedPips	Questa classe implementa l'algoritmo di Trailing Stop sulla base di punti fissi
CTrailingMA	Questa classe implementa l'algoritmo Trailing Stop in base ai valori dell' indicatore Moving Average
CTrailingNone	Una classe stub, non fa uso di nessun algoritmo Trailing Stop
CTrailingPSAR	Questa classe implementa l'algoritmo Trailing Stop in base ai valori dell' indicatore Parabolic SAR

Classi Money Management	Descrizione
CMoneyFixedLot	Una classe con un algoritmo basato sul trading con predefinita la grandezza del lotto fissa.
CMoneyFixedMargin	Una classe con un algoritmo basato sul trading con margine fisso predefinito.
CMoneyFixedRisk	Una classe con un algoritmo basato sul trading con rischio predefinito.

Classi Money Management	Descrizione
CMoneyNone	Una classe con un algoritmo basato sul trading con la grandezza minima del lotto consentita.
CMoneySizeOptimized	Una classe con un algoritmo basato sul trading con grandezza del lotto variabile a seconda dei risultati dei precedenti affari.

Classi base per Expert Advisors

Questa sezione contiene i dettagli tecnici di lavoro con le classi per la creazione e la sperimentazione di strategie di trading e descrizione dei componenti rilevanti della libreria standard MQL5.

L'uso di queste classi vi farà risparmiare tempo durante la creazione di strategie di trading.

La MQL5 standard Library (in termini di strategie di trading) viene inserita nella directory del terminale, nella cartella Include\Expert.

Classe	Descrizione
CExpertBase	Classe base per tutte le strategie di trading
CExpert	Classe base per l'Expert Advisor
CExpertSignal	Classe base per le classi Trading Signal (Segnali di Trading)
CExpertTrailing	Classe base per le classi Trailing Stop
CExpertMoney	Classe base per le classi di Money Management (Gestione del denaro)

CExpertBase

La classe CExpertBase è una classe di base per la classe CExpert e tutte le classi strategia di trading.

Descrizione

CExpertBase fornisce i dati e metodi, che sono comuni a tutti gli oggetti dell' Expert Advisor.

Dichiarazione

```
class CExpertBase : public CObject
```

Titolo

```
#include <Expert\ExpertBase.mqh>
```

Gerarchia di ereditarietà

CObject

CExpertBase

Discendenti diretti

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

Metodi della Classe

Metodi Pubblici:

Inizializzazione	
virtual Init	Metodo di inizializzazione istanza della classe
virtual ValidationSettings	Controlla le impostazioni
Parametri	
Symbol	Imposta il simbolo
Period	Imposta il timeframe
Magic	Imposta l'ID dell'Expert Advisor
Indicatori e Timeseries	
virtual SetPriceSeries	Imposta puntatori a timeseries esterne (serie di prezzi)
virtual SetOtherSeries	Imposta puntatori a timeseries esterne (serie non-prezzo)
virtual InitIndicators	Inizializza gli indicatori e timeseries
L'accesso ai Dati Protetti	
InitPhase	Ottiene l'attuale fase di inizializzazione dell'oggetto
TrendType	Imposta tipo di trend

Inizializzazione	
UsedSeries	Ottiene maschera di bit di timeseries usate
EveryTick	Imposta il flag "Ogni tick"
L'accesso a Timeseries	
Open	Ottiene l'elemento di Open delle timeseries per indice
High	Ottiene l'elemento di High delle timeseries per indice
Low	Ottiene l'elemento di Low delle timeseries per indice
Close	Ottiene l'elemento di Close delle timeseries per indice
Spread	Ottiene l'elemento di Spread delle timeseries per indice
Time	Ottiene l'elemento di Time delle timeseries per indice
TickVolume	Ottiene l'elemento di TickVolume delle timeseries per indice
RealVolume	Ottiene l'elemento di RealVolume delle timeseries per indice

Metodi Protetti

Inizializzazione di Timeseries	
InitOpen	Metodo inizializzazione timeseries Open
InitHigh	Metodo inizializzazione timeseries High
InitLow	Metodo inizializzazione timeseries Low
InitClose	Metodo inizializzazione timeseries Close
InitSpread	Metodo inizializzazione timeseries Spread
InitTime	Metodo inizializzazione timeseries Time
InitTickVolume	Metodo inizializzazione timeseries TickVolume
InitRealVolume	Metodo inizializzazione timeseries RealVolume
Metodi di Servizio	
virtual PriceLevelUnit	Ottiene l'unità di livello dei prezzi
virtual StartIndex	Ottiene l'indice della barra di partenza da analizzare
virtual CompareMagic	Confronta l'Expert Advisor ID con il valore specificato

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase

Ottiene l'attuale fase di inizializzazione dell'oggetto.

```
ENUM_INIT_PHASE  InitPhase()
```

Valore di ritorno

Attuale fase di inizializzazione dell'oggetto.

Nota

L'inizializzazione dell'oggetto consiste di diverse fasi:

1. Avvia l'inizializzazione.

- start - dopo la conclusione del costruttore
- finish - dopo il completamento con successo del metotoInit(...).
- allowed - chiamata del metodo Init(...)
- not allowed - chiamata del metodo ValidationSetting() ed altri metodi d'inizializzazione

2. Fase d'mpostazione dei parametri. In questa fase è necessario impostare tutti i parametri degli oggetti, utilizzati per la creazione degli indicatori.

- start - dopo il completamento con successo del metodo Init(...)
- finish - dopo il completamento con successo del metodo ValidationSettings()
- allowed - chiamata dei metodi Symbol(...) e Period(...)
- not allowed - chiamata dei metodi Init(...), SetPriceSeries(...), SetOtherSeries(...) and InitIndicators(...)

3. Controllo dei parametri.

- start - dopo il completamento con successo del metodo ValidationSettings()
- finish - dopo il completamento con successo del metodo InitIndicators(...)
- allowed - chiamata dei metodi Symbol(...), Period(...) ed InitIndicators(...)
- not allowed - chiamata di qualunque altro metodo d'inizializzazione

4. Fine dell' inizializzazione.

- start - dopo il completamento con successo del metodo InitIndicators(...)
- not allowed - chiamata dei metodi d'inizializzazione

TrendType

Imposta tipo di trend.

```
void  TrendType(
   M_TYPE_TREND    value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di tipo del trend.

Valore di ritorno

Nessuno.

UsedSeries

Ottiene maschera di bit di timeseries usate

```
int  UsedSeries()
```

Valore di ritorno

L'elenco dei timeseries utilizzato come maschera di bit.

Nota

Se il bit è impostato, il timeseries corrispondente è utilizzato, se non è impostato, il timeseries non viene utilizzato.

La corrispondenza bit-timeseries :

bit 0 - Open timeseries,
bit 1 - High timeseries,
bit 2 - Low timeseries,
bit 3 - Close timeseries,
bit 4 - Spread timeseries,
bit 5 - Time timeseries,
bit 6 - TickVolume timeseries,
bit 7 - RealVolume timeseries.

EveryTick

Imposta il flag "Ogni tick".

```
void  EveryTick(
   bool     value          // flag
   )
```

Parametri

value

[in] Nuovo valore del flag.

Valore di ritorno

Nessuno.

Nota

Se il flag non è impostato, il metodo di elaborazione viene chiamato solo alla nuova barra sul timefreme e simbolo lavoranti.

Open

Ottiene l'elemento di Open timeseries per indice.

```
double  Open(
   int     ind          // indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico dell'elemento Open timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

High

Ottiene l'elemento di High delle timeseries per indice

```
double  High(
   int     ind          // indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico dell'elemento High timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

Low

Ottiene l'elemento di Low delle timeseries per indice.

```
double  Low(
   int     ind          // indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico dell'elemento Low timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

Close

Ottiene l'elemento della timeseries Close per indice.

```
double  Close(
   int    ind         // indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico di Close timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

Spread

Ottiene l'elemento di Spread delle timeseries per indice

```
double  Spread(
   int    ind          // indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico dell'elemento Spread timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

Time

Ottiene l'elemento di Time delle timeseries per indice

```
datetime  Time(
   int    ind          // indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico dell'elemento Time timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

TickVolume

Ottiene l'elemento di TickVolume delle timeseries per indice

```
long  TickVolume(
   int     ind          // Indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico dell'elemento TickVolume timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

RealVolume

Ottiene l'elemento di RealVolume delle timeseries per indice

```
long  RealVolume(
   int    ind         // indice
   )
```

Parametri

ind

[in] Elemento dell'indice.

Valore di ritorno

In caso di successo, restituisce il valore numerico dell'elemento RealVolume timeseries con indice specificato, altrimenti restituisce EMPTY_VALUE.

Nota

EMPTY_VALUE viene restituito in due casi:

1. La TimeSeries non viene utilizzata (il bit corrispondente non è impostato).
2. L'Indice elemento è fuori dal range.

Init

Inizializza l'oggetto.

```
bool  Init(
   CSymbolInfo       symbol,    // simbolo
   ENUM_TIMEFRAMES   period,    // timeframe
   double            point      // punto
   )
```

Parametri

symbol

[in] Puntatore all'oggetto di tipo CSymbolInfo per l'accesso alle informazioni del simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES enumerazione).

point

[in] Il peso("weight") di 2/4-punti cifra.

Valore di ritorno

true in caso di successo, altrimenti false.

Symbol

Imposta il simbolo.

```
bool  Symbol(
   string    name     // simbolo
   )
```

Parametri

name

[in] Simbolo.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

L'impostazione del simbolo di lavoro è necessario se l'oggetto utilizza il simbolo diverso dal simbolo definito all'inizializzazione.

Period

Imposta il timeframe.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // timeframe
   )
```

Parametri

value

[in] Timeframe.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

L'impostazione del timeframe di lavoro è necessario se l'oggetto utilizza il timeframe diverso dal timeframe definito durante l'inizializzazione.

Magic

Imposta l'Expert Advisor ID.

```
void  Magic(
   ulong  value     // magic
   )
```

Parametri

value

[in] Expert Advisor ID.

Valore di ritorno

Nessuno.

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

SetPriceSeries

Imposta puntatori a serie di prezzi esterni.

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // puntatore
   CiHigh*    high,         // puntatore
   CiLow*     low,          // puntatore
   CiClose*   close         // puntatore
   )
```

Parametri

open

[in] Puntatore a Open timeseries.

high

[in] Puntatore a High timeseries.

low

[in] Puntatore a Low timeseries.

close

[in] Puntatore a Close timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

L'impostazione di puntatori a timeseries esterne (serie prezzo) è necessario se l'oggetto usa timeseries del simbolo e timeframe diversi dal simbolo e timeframe definiti in fase di inizializzazione.

SetOtherSeries

Imposta puntatori a serie esterne non-prezzo.

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // puntatore
   CiTime*         time,          // puntatore
   CiTickVolume*   tick_volume,   // puntatore
   CiRealVolume*   real_volume    // puntatore
   )
```

Parametri

spread

[in] Puntatore a Spread timeseries.

time

[in] Puntatore a Time timeseries.

tick_volume

[in] Puntatore a TickVolume timeseries.

real_volume

[in] Puntatore a RealVolume timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

L'impostazione di puntatori a timeseries esterne (serie non-prezzo) è necessario se l'oggetto usa timeseries del simbolo e timeframe diversi dal simbolo e timeframe definiti in fase di inizializzazione.

InitIndicators

Inizializza tutti gli indicatori e le timeseries.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

I timeseries vengono inizializzati solo se l'oggetto utilizza il simbolo o il timeframe, diversi dal simbolo o timeframe definiti in fase di inizializzazione.

InitOpen

Inizializza la Open timeseries.

```
bool  InitOpen(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Open timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

InitHigh

Initalizes la High timeseries.

```
bool  InitHigh(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

High timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

InitLow

Inizializza la Close timeseries.

```
bool  InitLow(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Low timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

InitClose

Inizializza la Close timeseries.

```
bool  InitClose(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Close timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

InitSpread

Inizializza la Spread timeseries.

```
bool  InitSpread(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Spread timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

InitTime

Inizializza la Time timeseries.

```
bool  InitTime(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Time timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

InitTickVolume

Initalizes le timeseries TickVolume.

```
bool  InitTickVolume(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

TickVolume timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

InitRealVolume

Inizializza la RealVolume timeseries.

```
bool  InitRealVolume(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

RealVolume timeseries viene inizializzato solo se l'Expert Advisor utilizza il simbolo/timeframe, diverso dal simbolo/timeframe definito in fase di inizializzazione (e la timeseries viene utilizzata ulteriormente).

PriceLevelUnit

Ottiene l'unità di livello dei prezzi.

```
virtual double  PriceLevelUnit()
```

Valore di ritorno

Il valore delle unità di livello di prezzi (Price Level).

Nota

Il metodo della classe base restituisce il "peso"(weight) di 2/4 punti cifre.

StartIndex

Ottiene l'indice della barra di partenza da analizzare

```
virtual int  StartIndex()
```

Valore di ritorno

L'indice della barra di partenza da analizzare

Nota

Il metodo restituisce 0 se la flag di analisi della barra corrente è impostata su true (analisi dalla barra corrente). Se il flag non è impostato, restituisce 1 (analisi dall'ultima barra completata).

CompareMagic

Confronta l'Expert Advisor ID (magic) con il valore specificato.

```
virtual bool  CompareMagic(
   ulong  magic    // valore da confrontare
   )
```

Parametri

magic

[in] Valore da confrontare.

Valore di ritorno

true se sono uguali, altrimenti false.

CExpert

CExpert è una classe base per strategie di trading. Essa è dotata di algoritmi per lavorare con serie storiche ed indicatori e una serie di metodi virtuali per la strategia di trading.

Come usarla:

1. Preparare un algoritmo della strategia;
2. Creare la propria classe, ereditata dalla classe CExpert;
3. Sovrascrivere i metodi virtuali nella tua classe con i propri algoritmi.

Descrizione

La classe CExpert è un insieme di metodi virtuali per l'implementazione di strategie di trading.

Nota

Una posizione viene riconosciuta come appartenente ad un Expert Advisor e gestita da esso in base alla coppia delle proprietà m_symbol ed m_magic. Nella modalità "hedging", più posizioni possono essere aperte per lo stesso simbolo, quindi il valore m_magic è importante.

Dichiarazione

```
class CExpert : public CExpertBase
```

Titolo

```
#include <Expert\Expert.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpert

Metodi della Classe

Inizializzazione	
Init	Metodo di inizializzazione istanza della classe
virtual InitSignal	Inizializza oggetto Trading Signal
virtual InitTrailing	Inizializza oggetto Trailing Stop
virtual InitMoney	Inizializza oggetto Money Management
virtual InitTrade	Inizializza oggetto Trade
virtual ValidationSettings	Controlla le impostazioni
virtual InitIndicators	Inizializza indicatori e timeseries
virtual InitParameters	Metodo di inizializzazione dei parametri

Inizializzazione	
virtual Deinit	Metodo di deinizializzazione istanza della classe
virtual DeinitSignal	Deinizializza l'oggetto Trading Signal
virtual DeinitTrailing	Deinizializza oggetto Trailing Stop
virtual DeinitMoney	Deinizializza oggetto Money Management
virtual DeinitTrade	Deinizializza oggetto Trade
virtual DeinitIndicators	Deinizializza Indicatori tecnici e TimeSeries
Parametri	
Magic	Imposta l'ID dell'Expert Advisor
MaxOrders	Ottiene/Imposta la quantità massima di ordini ammessi
OnTickProcess	Imposta un flag di procedere l'evento "onTick"
OnTradeProcess	Imposta un flag di procedere l'evento "OnTrade"
OnTimerProcess	Imposta un flag di procedere l'evento "OnTimer"
OnChartEventProcess	Imposta un flag di procedere l'evento "OnChartEvent"
OnBookEventProcess	Imposta un flag di procedere l'evento "OnBookEvent"
Metodi di Esecuzione degli eventi (Event Processing)	
OnTick	OnTick event handler
OnTrade	OnTrade event handler
OnTimer	OnTimer event handler
OnChartEvent	OnChartEvent event handler
OnBookEvent	OnBookEvent event handler
Metodi di aggiornamento	
Refresh	Aggiorna tutti i dati
Elabora	
Elabora	Algoritmo di elaborazione principale
Metodi di Ingresso nel Mercato (Entry Methods)	
CheckOpen	Controlla condizioni di apertura della posizione
CheckOpenLong	Controlla condizioni per aprire la posizioni long
CheckOpenShort	Controlla condizioni per aprire posizioni short
OpenLong	Apre una posizione long

Inizializzazione	
OpenShort	Apre una posizione short
Market Exit Methods	
CheckClose	Controlla condizioni per chiudere la posizione corrente
CheckCloseLong	Controlla condizioni per chiudere la posizione long
CheckCloseShort	Controlli condizioni per chiudere la posizione short
CloseAll	Chiude la posizione aperta ed elimina tutti gli ordini
Close	Chiude la posizione aperta
CloseLong	Chiude la posizione long
CloseShort	Chiude la posizione short
Metodi di Inversione della posizione	
CheckReverse	Controlla condizioni per invertire la posizione aperta
CheckReverseLong	Controlla condizioni per invertire posizione long
CheckReverseShort	Controlla condizioni per invertire posizione short
ReverseLong	Esegue operazione inversione della posizione long
ReverseShort	Esegue operazione inversione della posizione short
Metodi di Trailing Posizione/Ordine	
CheckTrailingStop	Controlla condizioni per modificare i parametri di posizione
CheckTrailingStopLong	Controlla condizioni di Trailing Stop della posizione long
CheckTrailingStopShort	Controlla condizioni di Trailing Stop della posizione short
TrailingStopLong	Esegue Trailing Stop per posizione long
TrailingStopShort	Esegue Trailing Stop per posizione short
CheckTrailingOrderLong	Controlla condizioni di Trailing Stop di ordine pendente Buy Limit/Stop
CheckTrailingOrderShort	Controlla condizioni di Trailing Stop di ordine pendente Sell Limit/Stop
TrailingOrderLong	Esegue Trailing Stop per ordine pendente Buy Limit/Stop
TrailingOrderShort	Esegue Trailing Stop per ordine pendente Sell Limit/Stop
Metodi di Eliminazione dell' Ordine	
CheckDeleteOrderLong	Controlla condizioni per eliminare ordine pendente Buy

Inizializzazione	
CheckDeleteOrderShort	Controlla condizioni per eliminare ordine pendente Sell
DeleteOrders	Elimina tutti gli ordini
DeleteOrder	Elimina ordine pendente Stop/Limit
DeleteOrderLong	Elimina ordine pendente Buy Stop/Limit
DeleteOrderShort	Elimina ordine pendente Sell Stop/Limit
Metodi di Volume di Trade	
LotOpenLong	Ottiene il volume di trade per operazione buy
LotOpenShort	Ottiene il volume di trade per operazione sell
LotReverse	Ottiene il volume trade per operazione inversione di posizione
Metodi di Storico di Trade	
PrepareHistoryDate	Imposta data di inizio per il monitoraggio storico di trading
HistoryPoint	Crea un checkpoint dello storico di trade (salva numero di posizioni, ordini, offerte e ordini storici)
CheckTradeState	Confronta lo stato attuale con quello salvato e chiama il corrispondente event handler
Event flags	
WaitEvent	Imposta il flag "attesa di evento"
NoWaitEvent	Reimposta il flag in attesa dell'evento
Metodi di Esecuzione dell'evento Trade(di trading)	
TradeEventPositionStopTake	Event handler dell'evento "Stop Loss/Take Profit della posizione innescato"
TradeEventOrderTriggered	Event handler dell'evento "Ordine Pendente innescato"
TradeEventPositionOpened	Event handler dell'evento "Posizione Aperta"
TradeEventPositionVolumeChanged	Event handler dell'evento "Volume della posizione cambiato"
TradeEventPositionModified	Event handler dell'evento "Posizione Modificata"
TradeEventPositionClosed	Event handler dell'evento "Posizione Chiusa"
TradeEventOrderPlaced	Event handler dell'evento "Ordine Pendente Piazzato"
TradeEventOrderModified	Event handler dell'evento "Ordine Pendente Modificato"
TradeEventOrderDeleted	Event handler dell'evento "Ordine Pendente Eliminato"
TradeEventNotIdentified	Event handler dell'evento non-identificato

Inizializzazione	
Metodi di servizio	
TimeframeAdd	Aggiunge un timeframe per tracciare
TimeframesFlags	Ottiene il flag che indica i timeframe con una nuova barra
SelectPosition	Seleziona la posizione con cui lavorare

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

Init

Metodo di inizializzazione istanza della classe

```
bool  Init(
   string              symbol,         // simbolo
   ENUM_TIMEFRAMES     period,         // timeframe
   bool                every_tick,     // flag
   ulong               magic           // magic
   )
```

Parametri

symbol

[in] Simbolo.

period

[in] Timeframe (ENUM_TIMEFRAMES enumerazione).

every_tick

[in] Flag.

magic

[in] Expert Advisor ID (Magic number).

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Se every_tick è impostato su true, il metodo Processing() viene chiamato ad ogni tick del simbolo lavorante. altrimenti, Processing() viene chiamato solo alla nuova barra del simbolo lavorante.

Magic

Imposta l'Expert Advisor ID (magic).

```
void  Magic(
   ulong  value     // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore dell' Expert Advisor ID.

Valore di ritorno

Nessuno.

Nota

Imposta il valore dell' Expert Advisor ID (magic) nelle seguenti classi: Trade, Signal, Money, Trailing.

Implementazione

```
//+------------------------------------------------------------------+
//| Imposta il magic number per oggetti ed i suoi oggetti dipendenti |
//| INPUT:  valore - nuovo valore del magic number.                  |
//| OUTPUT: no.                                                      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
void CExpert::Magic(ulong value)
  {
   if(m_trade!=NULL)     m_trade.SetExpertMagicNumber(value);
   if(m_signal!=NULL)    m_signal.Magic(value);
   if(m_money!=NULL)     m_money.Magic(value);
   if(m_trailing!=NULL) m_trailing.Magic(value);
//---
   CExpertBase::Magic(value);
  }
```

InitSignal

Inizializza oggetto Trading Signal.

```
virtual bool  InitSignal(
   CExpertSignal*      signal=NULL,         // puntatore
   )
```

Parametri

signal

[in] Puntatore alla classe CExpertSignal (o il suo erede).

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Se il segnale è NULL, verrà usata la classe CExpertSignal, e non fa nulla.

InitTrailing

Inizializza l'oggetto Trailing Stop.

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // pointer
   )
```

Parametri

trailing

[in] Puntatore all'oggetto della classe CExpertTrailing (o il suo erede).

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Se il trailing è NULL, verrà utilizzata la clase ExpertTrailing, e non fa nulla.

InitMoney

Inizializza l'oggetto Money Management.

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // puntatore
   )
```

Parametri

money

[in] Puntatore alla classe CExpertMoney (or il suo erede).

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Se money è NULL, la classe CExpertMoney verrà usata, utilizza il lotto minimo.

InitTrade

Inizializza l'oggetto Trade

```
virtual bool  InitTrade(
   ulong            magic,        // magic
   CExpertTrade*    trade=NULL    // puntatore
   )
```

Parametri

magic

[in] Expert Advisor ID (verrò usato nelle richieste di trade).

trade

[in] Puntatore all'oggetto CExpertTrade.

Valore di ritorno

true in caso di successo, altrimenti false.

Deinit

Metodo di deinizializzazione dell'istanza della classe.

```
virtual void Deinit()
```

Valore di ritorno

Nessuno.

OnTickProcess

Imposta un flag per procedere all'eventoOnTick.

```
void  OnTickOProcess(
   bool     value        // flag
   )
```

Parametri

value

[in] Flag per procedere all'evento OnTick.

Valore di ritorno

Nessuno.

Nota

Se il flag è true, l'evento OnTick viene processato, per default, il flag è impostato su false.

OnTradeProcess

Imposta una flag per procedere l'eventoOnTrade.

```
void  OnTradeProcess(
   bool     value        // flag
   )
```

Parametri

value

[in] Flag per procedere all'evento OnTrade.

Valore di ritorno

Nessuno.

Nota

Se il flag è true, l'evento OnTrade viene processato, per default, il flag viene impostato su false.

OnTimerProcess

Imposta una flag per procedere l'evento OnTimer.

```
void  OnTimerProcess(
   bool     value        // flag
   )
```

Parametri

value

[in] Flag per procedere all' eventoOnTimer.

Valore di ritorno

Nessuno.

Nota

Se il flag è true, viene processato l'evento OnTimer, per impostazione predefinita il flag è impostato su false.

OnChartEventProcess

Imposta una flag per procedere l'evento OnChartEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Parametri

value

[in] Flag per procedere all' eventoOnChartEvent.

Valore di ritorno

Nessuno.

Nota

Se il flag è true, l'evento OnChartEvent è di procedere, per default, il flag viene impostato su false.

OnBookEventProcess

Imposta una flag per procedere l'evento OnBookEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Parametri

value

[in] Flag per procedere all' eventoOnBookEvent.

Valore di ritorno

Nessuno.

Nota

Se il flag è true, l'evento OnBookEvent viene processato, per default, il flag viene impostato su false.

MaxOrders (Metodo Get)

Ottiene la massima quantità di ordini consentiti.

```
int  MaxOrders()
```

Valore di ritorno

Ammontare massimo di ordini consentiti.

MaxOrders (Metodo Set)

Imposta la quantità massima di ordini consentiti.

```
void  MaxOrders(
   int      max_orders        // nuovo valore
   )
```

Parametri

max_orders

[in] Nuovo valore della quantità massima di ordini consentiti.

Valore di ritorno

Nessuno.

Nota

Per default, l'importo massimo degli ordini consentito è = 1.

Signal

Ottiene il puntatore all'oggetto Trade Signal.

```
CExpertSignal*  Signal() const
```

Valore di ritorno

Puntatore all'oggetto Trade Signal.

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Controlla anche le impostazioni di tutti gli oggetti Expert Advisor.

InitIndicators

Inizializza tutti gli indicatori e timeseries.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Le timeseries vengono inizializzate se l'oggetto utilizza un simbolo o timeframe diverso da quelli definiti nell' inizializzazione. Chiama conseguenzialmente InitIndicators() metodi degli oggetti virtuali di trading segnal, trailing stop e money management.

OnTick

Event handler dell'evento OnTick.

```
virtual void  OnTick()
```

Valore di ritorno

Nessuno.

OnTrade

Event handler dell'evento OnTrade.

```
virtual void  OnTrade()
```

Valore di ritorno

Nessuno.

OnTimer

Event handler dell'evento OnTimer.

```
virtual void  OnTimer()
```

Valore di ritorno

Nessuno.

OnChartEvent

Event handler dell'evento OnChartEvent.

```
virtual void  OnChartEvent(
   const int     id,       // id evento
   const long&   lparam,   // parametro long
   const double  dparam,   // parametro double
   const string  sparam    // parametro string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Evento parametro di tipo long.

dparam

[in] Evento parametro di tipo double.

sparam

[in] Evento parametro di tipo string.

Valore di ritorno

Nessuno.

OnBookEvent

Event handler dell'evento OnBookEvent.

```
virtual void  OnBookEvent(
   const string&     symbol       // simbolo
)
```

Parametri

symbol

[in] Simbolo dell'evento OnBookEvent.

Valore di ritorno

Nessuno.

InitParameters

Inizializza i parametri dell' Expert Advisor.

```
virtual bool  InitParameters()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

La funzione InitParameters() della classe base CExpert non fa nulla e restituisce sempre true.

DeinitTrade

Deinizializza l'oggetto Trade

```
virtual void  DeinitTrade()
```

Valore di ritorno

Nessuno.

DeinitSignal

Deinizializza l'oggetto Signal

```
virtual void  DeinitSignal()
```

Valore di ritorno

Nessuno.

DeinitTrailing

Deinizializza l'oggetto Trailing

```
virtual void  DeinitTrailing()
```

Valore di ritorno

Nessuno.

DeinitMoney

Deinizializza oggetto Money Management

```
virtual void  DeinitMoney()
```

Valore di ritorno

Nessuno.

DeinitIndicators

Deinizializza tutti gli indicatori e le time series.

```
virtual void  DeinitIndicators()
```

Valore di ritorno

Nessuno.

Nota

Deinizializza in anche tutti gli indicatori e le timeseries di tutti gli oggetti ausiliari.

Refresh

Aggiorna tutti i dati.

```
virtual bool  Refresh()
```

Valore di ritorno

true se è necessaria l'ulteriore elaborazione di ticks, altrimenti false.

Nota

Esso permette di determinare la necessità di elaborazione dei tick. Se necessario, aggiorna tutte le quotazioni ed i dati di timeseries ed indicatori e restituisce true.

Implementazione

```
//+------------------------------------------------------------------+
//| Aggiorna i dati per l'elaborazione                               |
//| INPUT:  no.                                                      |
//| OUTPUT: true-in caso di successo, altrimenti false.              |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- aggiora i tassi
   if(!m_symbol.RefreshRates()) return(false);
//--- controlla la necessità d'elaborare
   TimeToStruct(m_symbol.Time(),time);
   if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
      if((m_period_flags & TimeframesFlags(time))==0) return(false);
   m_last_tick_time=time;
//--- aggiorna indicatori
   m_indicators.Refresh();
//--- ok
   return(true);
  }
```

Elabora

Algoritmo di elaborazione principale.

```
virtual bool  Processing()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Fa le seguenti operazioni:

1. Verifica la presenza della posizione aperta sul simbolo. Se non ci sono posizioni aperte, salta i passaggi №2, №3 and №4.
2. Controlli condizioni per invertire la posizione aperta (metodo CheckReverse()). Se la posizione è stata "invertita", esce.
3. Controlli condizioni per chiudere la posizione (metodo CheckClose ()). Se la posizione è stata chiusa, salta il passaggio №4.
4. Verifica le condizioni per modificare i parametri della posizione (metodo CheckTrailingStop()). Se i parametri della posizione sono stati modificati, esce.
5. Controlla la presenza di ordini pendenti sul simbolo. Se non c'è alcun ordine pendete, va al passaggio №9.
6. Controlla condizioni per eliminare ordine (CheckDeleteOrderLong() per ordini pendenti buy o CheckDeleteOrderShort() per ordini pendenti sell). Se l'ordine è stato eliminato, va al passaggio №9.
7. Controlla condizioni per modificare i parametri dell'ordine pendente (CheckTrailingOrderLong() per ordini buy oppure CheckTrailingOrderShort() per ordini sell). Se i parametri dell'ordine sono stati modificati, esce.
8. Exit.
9. Controlli condizioni per aprire posizione (metodo CheckOpen()).

Se si desidera implementare il proprio algoritmo, hai bisogno di eseguire l'override del metodo Processing() della classe erede.

Implementazione

```
//+------------------------------------------------------------------+
//| Funzione principale                                              |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se una qualunque operazione di trade è stata elaborata, false altrime
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- controlal se ci sono posizioni aperte
   if(m_position.Select(m_symbol.Name()))
     {
      //--- posizione aperta è disponibile
      //--- controlla la possibilità di invertibilità posizione
      if(CheckReverse()) return(true);
      //--- controlla la possibilità di chisura della posizione/eliminazione ordini pe
      if(!CheckClose())
        {
         //--- controlla la possibilità di modifica della posizione
         if(CheckTrailingStop()) return(true);
         //--- ritorno senza operazioni
         return(false);
        }
     }
//--- controlla se ci sono oridini pendenti piazzati
   int total=OrdersTotal();
   if(total!=0)
     {
      for(int i=total-1;i>=0;i--)
        {
         m_order.SelectByIndex(i);
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()==ORDER_TY
           {
            //--- controlla la possibilità di eliminare un ordine pendente di buy
            if(CheckDeleteOrderLong()) return(true);
            //--- controlla la possibilità di modifica di un ordine pendente di buy
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- controlla la possibilità di eliminare un ordine pendente sell
            if(CheckDeleteOrderShort()) return(true);
            //--- controlla la possibilità di modifica di un ordine pendente sell
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- ritorno senza operazioni
         return(false);
        }
     }
//--- controlla la possibilità di apertura di una posizione/impostazione di un ordine
   if(CheckOpen()) return(true);
//--- ritorno senza operazioni
   return(false);
  }
```

SelectPosition

Seleziona la posizione con cui lavorare.

```
void  SelectPosition()
```

Valore di ritorno

No.

Implementazione

```
//+------------------------------------------------------------------+
//| Position select                                                  |
//| INPUT:  no.                                                      |
//| OUTPUT: no.                                                      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::SelectPosition(void)
  {
   bool res=false;
//---
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      res=m_position.SelectByMagic(m_symbol.Name(),m_magic);
   else
      res=m_position.Select(m_symbol.Name());
//---
   return(res);
  }
```

CheckOpen

Controlli condizioni per aprire una posizione.

```
virtual bool  CheckOpen()
```

Valore di ritorno

true se è stata eseguita qualsiasi operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per aprire psozioni long (CheckOpenLong()) e short (CheckOpenShort()).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per apertura posizione o impostazione ordini limit/stop
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- ritorno senza operazioni
   return(false);
  }
```

CheckOpenLong

Controlla condizioni per aprire la posizioni long

```
virtual bool  CheckOpenLong()
```

Valore di ritorno

true se è stata eseguita qualsiasi operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per aprire una posizione long (CheckOpenLong() metodo dell'oggetto Signal) ed apre una posizione long (metodo OpenLong()) se necessario.

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per aperturara posizione long impostazione ordini limit/stop
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- controlla il segnale per operazioni di ingresso long
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- ritorno senza operazioni
   return(false);
  }
```

CheckOpenShort

Controlla condizioni per aprire posizioni short

```
virtual bool  CheckOpenShort()
```

Valore di ritorno

true se è stata eseguita qualsiasi operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per aprire una posizione short (CheckOpenShort() metodo dell'oggetto Signal) ed apre una posizione short (metodo OpenShort()) se necessario.

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per aperturara posizione short impostazione ordini limit/stop
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- controlla il segnale per operazioni di ingresso short
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- ritorno senza operazioni
   return(false);
  }
```

OpenLong

Apre una posizione long.

```
virtual bool  OpenLong(
   double    price,    // prezzo
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

tp

[in] prezzo Take Profit.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Ottiene il volume di trade (metodo LotOpenLong(...)) ed apre una posizione long (metodo Buy() dell'oggetto Tradce) se il volume di trading non è uguale a 0.

Implementazione

```
//+------------------------------------------------------------------+
//| Apertura posizione Long o impostazione ordine limit/stop         |
//| INPUT:  price - prezzo,                                          |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.  |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- ottiene il lotto per l'apertura
   double lot=LotOpenLong(price,sl);
//--- controlla il lotto per l'apertura
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

OpenShort

Apre una posizione short.

```
virtual bool  OpenShort(
   double    price,    // prezzo
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

tp

[in] prezzo Take Profit.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Ottiene il volume di trade (metodo LotOpenShort(...)) ed apre una posizione short (metodo Sell() dell'oggetto Tradce) se il volume di trading non è uguale a 0.

Implementazione

```
//+------------------------------------------------------------------+
//| Apertura posizione Short o impostazione ordine limit/stop        |
//| INPUT:  price - prezzo,                                          |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: true-se l'operazione di trade ha avuto successo, altrimenti false..  |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- ottiene lotto per apertura
   double lot=LotOpenShort(price,sl);
//--- controllo lotto per apertura
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

CheckReverse

Controlli condizioni per invertire la posizione aperta.

```
virtual bool  CheckReverse()
```

Valore di ritorno

true se è stata eseguita qualsiasi operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per invertire posizioni long (CheckReverseLong()) e short (CheckReverseShort()).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per inversione posizione                               |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- controlla la possibilità di invertire la posizione long
      if(CheckReverseLong())  return(true);
     }
   else
      //--- controlla la possibilità di invertire la posizione short
      if(CheckReverseShort()) return(true);
//--- ritorno senza operazioni
   return(false);
  }
```

CheckReverseLong

Controlla condizioni per invertire la posizione long.

```
virtual bool  CheckReverseLong()
```

Valore di ritorno

true se è stata eseguita qualsiasi operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per invertire la posizione long (CheckReverseLong() metodo dell'oggetto Signal) ed esegue l'operazione inversa della posizione long corrente (metodo ReverseLong(...)) se necessario.

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per inversione posizione long                          |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- controlla il segnale per l'operazione di inversione di long
   if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
//--- ritorno senza operazioni
   return(false);
  }
```

CheckReverseShort

Controlla condizioni per invertire la posizione short.

```
virtual bool  CheckReverseLong()
```

Valore di ritorno

true se è stata eseguita qualsiasi operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per invertire la posizione short (CheckReverseShort() metodo dell'oggetto Signal) ed esegue l'operazione inversa della posizione short corrente (metodo ReverseShort(...)) se necessario.

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per inversione posizione short                         |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- controlla il segnale per l'operazione di inversione di short
   if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl
//--- ritorno senza operazioni
   return(false);
  }
```

ReverseLong

Esegue un'operazione di inversione posizione long.

```
virtual bool  ReverseLong(
   double    price,    // prezzo
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

tp

[in] prezzo Take Profit.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Ottiene il volume di inversione posizione (metodo LotReverse()) ed inverte una posizione long (Sell() metodo dell'oggetto Trade) se il volume di trade non è uguale a 0.

In modalità "hedging" di accounting della posizione, viene eseguita l' inversione della posizione così come la chiusura della posizione esistente e l'apertura di una nuova opposta con il volume rimanente.

Implementazione

```
//+------------------------------------------------------------------+
//| Inversione posizione Long                                        |
//| INPUT:  price - prezzo,                                          |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE)
      return(false);
//--- ottiene il lotto d'inversione
   double lot=LotReverse(sl);
//--- check lot
   if(lot==0.0)
      return(false);
//---
   bool result=true;
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- first close existing position
      lot-=m_position.Volume();
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
     }
   if(result)
      result=m_trade.Sell(lot,price,sl,tp);
//---
   return(result);
  }
```

ReverseShort

Esegue operazione inversione della posizione short

```
virtual bool  ReverseShort(
   double    price,    // prezzo
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

tp

[in] prezzo Take Profit.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Ottiene il volume di inversione posizione (metodo LotReverse()) e fa un'operazione di trade di inversione posizione short (Buy() metodo dell'oggetto Trade) se il volume di trade non è uguale a 0.

In modalità "hedging" di accounting della posizione, viene eseguita l' inversione della posizione così come la chiusura della posizione esistente e l'apertura di una nuova opposta con il volume rimanente.

Implementazione

```
//+------------------------------------------------------------------+
//| Inversione posizione Short                                       |
//| INPUT:  price - price,                                           |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE)
      return(false);
//--- ottiene il lotto d'inversione
   double lot=LotReverse(sl);
//--- check lot
   if(lot==0.0)
      return(false);
//---
   bool result=true;
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- first close existing position
      lot-=m_position.Volume();
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
     }
   if(result)
      result=m_trade.Sell(lot,price,sl,tp);
//---
   return(result);
  }
```

CheckClose

Controlla condizioni per chiudere la posizione.

```
virtual bool  CheckClose()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

1. Controlla le condizioni di Stop Out dell' Expert Advisor (CheckClose () metodo dell'oggetto di gestione del denaro) . Se la condizione è soddisfatta, chiude la posizione ed elimina tutti gli ordini (CloseAll()) ed esce.
2. Controlla le condizioni per chiudere una posizione long o short (metodi CheckCloseLong() o CheckCloseShort()) e se la posizione è chiusa, elimina tutti gli ordini (metodo DeleteOrders()).

Implementazione

```
//+------------------------------------------------------------------+
//| Controlla per la posizione chiusa o eliminazione di ordini limit/stop |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- la posizione dev'essere selezionata prima della chiamata
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- controllo del tipo di posizione
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- controllo della possibilità di chiusura della posizione long / eliminazior
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- controllo della possibilità di chiusura della posizione short / eliminazio
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- ritorno senza operazioni
   return(false);
  }
```

CheckCloseLong

Controlla le condizioni per chiudere la posizione long.

```
virtual bool  CheckCloseLong()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per chiudere una posizione long (CheckCloseLong() metodo dell'oggetto Signal) e se sono soddisfatti, viene chiusa la posizione aperta (metodo CloseLong(...)).

Implementazione

```
//+------------------------------------------------------------------+
//| Controlla per la chiusura della posizionelong o eliminazione di ordini limit/stop
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- controlla per l'operazione chiusura long
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- ritorno senza operazioni
   return(false);
  }
```

CheckCloseShort

Controlla le condizioni per chiudere la posizione short.

```
virtual bool  CheckCloseShort()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni per chiudere la posizione short (CheckCloseShort() metodo dell'oggetto Signal) e se sono soddisfatte, chiude la posizione (metoto CloseShort()).

Implementazione

```
//+------------------------------------------------------------------+
//| Controlla per la chiusura posizione shorto eliminazione ordini limit/stop |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//--- controlla per operazioni di chiusura short
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- ritorno senza operazioni
   return(false);
  }
```

CloseAll

Esegue chiusura parziale o totale della posizione.

```
virtual bool  CloseAll(
   double    lot    // lotto
   )
```

Parametri

lot

[in] Numero di lotti per ridurre la posizione.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Nella modalità "netting", una posizione viene chiusa con i metodi CExpertTrade::Buy o CExpertTrade::Sell. Nella modalità "hedging", viene utilizzato il metodo CTrade::PositionCloseByTicket. Il metodo DeleteOrders() viene utilizzato per eliminare gli ordini.

Implementazione

```
//+------------------------------------------------------------------+
//| Chiusura posizione ed eliminazione ordini                        |
//| INPUT:  lotti - volume da chiudere.                              |
//| OUTPUT: true-se l'operazione di trade è stata eseguita, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result=false;
//--- controllo per chiusura operazioni
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
     {
      if(m_position.PositionType()==POSITION_TYPE_BUY)
         result=m_trade.Sell(lot,0,0,0);
      else
         result=m_trade.Buy(lot,0,0,0);
     }
   result|=DeleteOrders();
//---
   return(result);
  }
```

Close

Chiude posizione aperta.

```
virtual bool  Close()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Chiude la posizione (PositionClose() medoto dell'oggetto della classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Chiude Posizione                                                 |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

CloseLong

Chiude la posizione long.

```
virtual bool  CloseLong(
   double    price    // prezzo
   )
```

Parametri

price

[in] Prezzo.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Nella modalità "netting", una posizione viene chiusa con i metodi CExpertTrade::Buy o CExpertTrade::Sell. Nella modalità "hedging", viene utilizzato il metodo CTrade::PositionCloseByTicket.

Implementazione

```
//+------------------------------------------------------------------+
//| Chiusura Posizione Long                                          |
//| INPUT:  prezzo - prezzo per la chiusura.                         |
//| OUTPUT: true-se l'operazione di trade è stata eseguita, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   bool result=false;
//---
   if(price==EMPTY_VALUE)
      return(false);
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
      result=m_trade.Sell(m_position.Volume(),price,0,0);
//---
   return(result);
  }
```

CloseShort

Chiude la posizione short.

```
virtual bool  CloseShort(
   double    price    // prezzo
   )
```

Parametri

price

[in] Prezzo.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Nella modalità "netting", una posizione viene chiusa con i metodi CExpertTrade::Buy o CExpertTrade::Sell. Nella modalità "hedging", viene utilizzato il metodo CTrade::PositionCloseByTicket.

Implementazione

```
//+------------------------------------------------------------------+
//| ChiusuraPosizione Short                                          |
//| INPUT:  prezzo - prezzo per la chiusura.                         |
//| OUTPUT: true-se l'operazione di trade ha avuto successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   bool result=false;
//---
   if(price==EMPTY_VALUE)
      return(false);
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
      result=m_trade.Buy(m_position.Volume(),price,0,0);
//---
   return(result);
  }
```

CheckTrailingStop

Esso controlla le condizioni di Trailing Stop della posizione aperta.

```
virtual bool  CheckTrailingStop()
```

Valore di ritorno

true se è stata eseguita qualsiasi operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni di Trailing Stop della posizione aperta (CheckTrailingStopLong() o CheckTrailingStopShort() per le posizioni long e short).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per trailing stop/profitto posizione                   |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- la posizione dev'essere selezionata prima della chiamata
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- controlla la possibilità di modificare la posizione long
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- controlla la possibilità di modifica della posizione short
      if(CheckTrailingStopShort()) return(true);
     }
//--- ritorno senza operazioni
   return(false);
  }
```

CheckTrailingStopLong

Esso controlla le condizioni di Trailing Stop della posizione long aperta.

```
virtual bool  CheckTrailingStopLong()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni Trailing Stop della posizione long aperta (CheckTrailingStopLong(...) metodo dell'oggetto Expert Trailing). Se le condizioni sono soddisfatte, esso modifica i parametri della posizione (metodo TrailingStopLong(...)).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per trailing stop/profitto posizione long              |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- controllo per operazioni long trailing stop
   if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- operazioni long trailing stop
      return(TrailingStopLong(sl,tp));
     }
//--- ritorno senza operazioni
   return(false);
  }
```

CheckTrailingStopShort

Esso controlla le condizioni di Trailing Stop della posizione short aperta.

```
virtual bool  CheckTrailingStopShort()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni Trailing Stop della posizione short aperta (CheckTrailingStopShort(...) metodo dell'oggetto Expert Trailing). Se le condizioni sono soddisfatte, esso modifica i parametri della posizione (metodo TrailingStopShort(...)).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per trailing stop/profitto posizione short             |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- controllo per operazioni short trailing stop
   if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- controllo per operazioni short trailing stop
      return(TrailingStopShort(sl,tp));
     }
//--- tirono senza operazioni
   return(false);
  }
```

TrailingStopLong

Modifica parametri della posizione long aperta.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Parametri

sl

[in] Prezzo Stop Loss.

tp

[in] prezzo Take Profit.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

La funzione di modifica dei parametri della posizione long aperta (metodo PositionModify(...) oggetto classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Trailing stop/profit posizione long                              |
//| INPUT:  sl - nuovo stop loss,                                    |
//|         tp - nuovo take profit.                                  |
//| OUTPUT: true-se l'operazione di trade ha successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

TrailingStopShort

Modifica i parametri della posizione short aperta.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Parametri

sl

[in] Prezzo Stop Loss.

tp

[in] Prezzo Take Profit.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

La funzione modifica i parametri della posizione short aperta (metodo PositionModify(...) oggetto della classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Trailing stop/profit posizione short                             |
//| INPUT:  sl - nuovo stop loss,                                    |
//|         tp - nuovo take profit.                                  |
//| OUTPUT: true-se l'operazione di trade ha successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

CheckTrailingOrderLong

Controlla condizioni di Trailing Stop di ordini pendenti Buy Limit/Stop.

```
virtual bool  CheckTrailingOrderLong()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni Trailing Stop per ordini pendenti buy limit/stop (CheckTrailingOrderLong() metodo dell'oggetto Trade Signals) e modifica i parametri dell'ordine se necessario (metodo TrailingOrderLong(...)).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per il traling di ordini limit/stop                    |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- controlla la possibilità di modificare l'ordine long
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- ritorno senza operazioni
   return(false);
  }
```

CheckTrailingOrderShort

Esso controlla le condizioni Trailing Stop di oprdini pendenti Sell Limit/Stop.

```
virtual bool  CheckTrailingOrderShort()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla le condizioni Trailing Stop per ordini sell limit/stop (CheckTrailingOrderShort() metodo dell'oggetto Trade Signals) e modifica i parametri dell'ordine se necessario (metodo TrailingOrderShort()).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per trailing ordini limit/stop short                   |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- controlla la possibilità di modificare l'ordine short
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- ritorno senza operazioni
   return(false);
  }
```

TrailingOrderLong

Esso modifica i parametri dell' ordine Buy Limit/Stop pendente.

```
virtual bool  TrailingOrderLong(
   double    delta    // delta
   )
```

Parametri

delta

[in] Prezzo delta.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso modifica i parametri dell' ordine pendente Buy Limit/Stop (metodo OrderModify(...) oggetto classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Trailing long limit/stop order                                   |
//| INPUT:  delta - cambio prezzo.                                   |
//| OUTPUT: true-se l'operazione ha avuto successo, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderLong(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modifica l'ordine long
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

TrailingOrderShort

Esso modifica i parametri dell' ordine Sell Limit/Stop pendente.

```
virtual bool  TrailingOrderShort(
   double    delta    // delta
   )
```

Parametri

delta

[in] Prezzo delta.

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso modifica i parametri dell' ordine pendente Sell Limit/Stop (metodo OrderModify(...) oggetto classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Trailing short limit/stop order                                  |
//| INPUT:  delta - price change.                                    |
//| OUTPUT: true-se l'operazione ha avuto successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderShort(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modifica dell'ordine short
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

CheckDeleteOrderLong

Esso controlla le condizioni per eliminare ordini pendenti Buy Limit/Stop .

```
virtual bool  CheckDeleteOrderLong()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla l'orario di espirazione(scadenza) dell'ordine. Esso controlla le condizioni per eliminare Buy: ordine pendente Limite/Stop (CheckCloseLong(...) metodo dell'oggetto Signal della classe) ed elimina l'ordine se la condizione è soddisfatta (metodo DeleteOrderLong()).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per eliminazione ordini di limit/stop                  |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- controlla la possibilità di eliminazione dell'ordine long
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- ritorno senza operazioni
   return(false);
  }
```

CheckDeleteOrderShort

Esso controlla le condizioni per eliminare ordini pendenti Sell Limit/Stop .

```
virtual bool  CheckDeleteOrderShort()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Esso controlla l'orario di espirazione(scadenza) dell'ordine. Si controlla le condizioni per eliminare l'ordine pendente Sell Limit/Stop (metodo CheckCloseShort(...) dell' oggetto Signal della classe) ed elimina l'ordine se la condizione è soddisfatta (metodo DeleteOrderShort ()).

Implementazione

```
//+------------------------------------------------------------------+
//| Controllo per l'eliminazione di ordini short limit/stop
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade è stata elaborata, altrimenti false.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- controlla la possibilità di eliminazione dell'ordine short
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- ritorno senza operazioni
   return(false);
  }
```

DeleteOrders

Elimina tutti gli ordini.

```
virtual bool  DeleteOrders()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Elimina tutti gli ordini (DeleteOrder() per tutti gli ordini).

Implementazione

```
//+------------------------------------------------------------------+
//| Elimina tutti gli ordini limit/stop                              |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade ha avuto successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrders()
  {
   bool result=false;
   int  total=OrdersTotal();
//---
   for(int i=total-1;i>=0;i--)
     {
      if(m_order.Select(OrderGetTicket(i)))
        {
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         result|=DeleteOrder();
        }
     }
//---
   return(result);
  }
```

DeleteOrder

Elimina ordine pendente Stop/Limit

```
virtual bool  DeleteOrder()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Elimina l'ordine pendente Limit/Stop (OrderDelete(...) metodo dell'oggetto della classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Elimina ordine limit/stop                                        |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade ha avuto successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

DeleteOrderLong

Elimina l'ordine Buy Limit/Stop pendente.

```
virtual bool  DeleteOrderLong()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Elimina l'ordine pendente Buy Limit/Stop (OrderDelete(...) metodo dell'oggetto della classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Elimina l'ordine pendente limit/stop                             |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade ha avuto successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

DeleteOrderShort

Elimina l'ordine pendente Sell Limit/Stop.

```
virtual bool  DeleteOrderShort()
```

Valore di ritorno

true se è stata eseguita un'operazione di trade, altrimenti false.

Nota

Elimina l'ordine pendente Sell Limit/Stop (OrderDelete(...) metodo dell'oggetto della classe CTrade).

Implementazione

```
//+------------------------------------------------------------------+
//| Elimina l'ordine pendete short limit/stop                        |
//| INPUT:  no.                                                      |
//| OUTPUT: true-se l'operazione di trade ha avuto successo, altrimenti false.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

LotOpenLong

Ottiene il volume di trade per operazioni buy.

```
double  LotOpenLong(
   double    price,   // prezzo
   double    sl       // Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

Volume di Trade (in lotti) per l'operazione buy.

Nota

Ottiene il volume di trade per l'operazione buy (CheckOpenLong(...) metodo dell'oggetto money management).

Implementazione

```
//+------------------------------------------------------------------+
//| Metodo di ottenimento del lotto per l'operazione long aperta.    |
//| INPUT:  price - prezzo,                                          |
//|         sl    - stop loss.                                       |
//| OUTPUT: lotto per open.                                          |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

LotOpenShort

Ottiene il volume di trade per l'operazione sell

```
double  LotOpenShort(
   double    price,   // prezzo
   double    sl       // Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

Volume di Trade (in lotti) per l'operazione sell.

Nota

Ottiene il volume di trade per l'operazione sell (CheckOpenShort(...) metodo dell'oggetto money management).

Implementazione

```
//+------------------------------------------------------------------+
//| Metodo di ottenimento del lotto per l'operazione short aperta..  |
//| INPUT:  price - prezzo,                                          |
//|         sl    - stop loss.                                       |
//| OUTPUT: lotto per open.                                          |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

LotReverse

Ottiene il volume trade per operazione inversione di posizione

```
double  LotReverse(
   double    sl       // Stop Loss
   )
```

Parametri

sl

[in] Prezzo Stop Loss.

Valore di ritorno

Volume di Trade (in lotti) per l'operazione di inversione posizione.

Nota

Ottiene il volume di trade per l'operazione di inversione posizione (CheckReverse(...) metodo dell'oggetto money management).

Implementazione

```
//+------------------------------------------------------------------+
//| Metodo di ottenimento del lotto per l'inversione della posizione.
//| INPUT:  sl - stop loss.                                          |
//| OUTPUT: lotto per aprire.                                        |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

PrepareHistoryDate

Imposta la data d'inizio per il tracking della cronistoria di trade.

```
void  PrepareHistoryDate()
```

Nota

Il tracking del cronistorico di trade è impostato dall'inizio del mese (ma non meno di un giorno).

HistoryPoint

Crea un checkpoint della cronistoria di trade (salva un numero di posizioni, ordini, affari ed ordini cronistorici).

```
void  HistoryPoint(
   bool   from_check_trade=false   // flag
   )
```

Parametri

from_check_trade=false

[in] Flag per evitare la ricorsione.

Nota

Salva l'ammontare di posizioni, ordini, affari ed ordini cronistorici.

CheckTradeState

Confronta lo stato attuale con quello salvato e chiama il corrispondente event handler.

```
bool  CheckTradeState()
```

Valore di ritorno

true se l'evento è stata gestito(handled), in caso contrario - false.

Nota

Esso controlla il numero di posizioni, ordini, affari ed ordini cronistorici attraverso il confronto con i valori salvati dal metodo HistoryPoint(). Se la cronistoria di trade è cambiata, chiama il corrispondente virtual event handler.

WaitEvent

Imposta il flag di attesa di evento

```
void  WaitEvent(
   ENUM_TRADE_EVENTS    event          // flag
   )
```

Parametri

event

[in] Flag con eventi da impostare (enumerazione ENUM_TRADE_EVENTS).

Valore di ritorno

Nessuno.

Event flags

```
//--- flags di eventi attesi
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // niente eventi attesi
   TRADE_EVENT_POSITION_OPEN         =0x1,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_PLACE           =0x20,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_DELETE          =0x80,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // flag per l'aspettarsi dell'evento
  };
```

NoWaitEvent

Reimposta la flag di attesa dell'evento.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS    event         // flag
   )
```

Parametri

event

[in] Flag con eventi da azzerare (enumerazione ENUM_TRADE_EVENTS).

Valore di ritorno

Nessuno.

Event flags

```
//--- flags di eventi attesi
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // niente eventi attesi
   TRADE_EVENT_POSITION_OPEN         =0x1,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // flag per l'aspettarsi dell'evento
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_PLACE           =0x20,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_DELETE          =0x80,      // flag per l'aspettarsi dell'evento
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // flag per l'aspettarsi dell'evento
  };
```

TradeEventPositionStopTake

Event handler dell'evento "Stop Loss/Take Profit della posizione innescato".

```
virtual bool  TradeEventPositionStopTake()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventOrderTriggered

Event handler dell'evento "Ordine Pendente innescato".

```
virtual bool  TradeEventOrderTriggered()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventPositionOpened

Event handler dell'evento "Posizione Aperta".

```
virtual bool  TradeEventPositionOpened()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventPositionVolumeChanged

Event handler dell'evento "Volume della posizione cambiato".

```
virtual bool  TradeEventPositionVolumeChanged()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventPositionModified

Event handler dell'evento "Posizione Modificata".

```
virtual bool  TradeEventPositionModified()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventPositionClosed

Event handler dell'evento "Posizione Chiusa".

```
virtual bool  TradeEventPositionClosed()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventOrderPlaced

Event handler dell'evento "Ordine Pendente Piazzato".

```
virtual bool  TradeEventOrderPlaced()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventOrderModified

Event handler dell'evento "Ordine Pendente Modificato".

```
virtual bool  TradeEventOrderModified()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventOrderDeleted

Event handler dell'evento "Ordine Pendente Eliminato".

```
virtual bool  TradeEventOrderDeleted()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

TradeEventNotIdentified

Event handler dell'evento non identificato.

```
virtual bool  TradeEventNotIdentified()
```

Valore di ritorno

The CExpert il metodo della classe non fa nulla e restituisce sempre true.

Nota

Si noti che possono arrivare svariati eventi di trade, in questi casi è difficile identificarli.

TimeframeAdd

Aggiunge un timeframe per il monitoraggio.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period         // timeframe
   )
```

Parametri

period

[in] Timeframe (ENUM_TIMEFRAMES enumeration).

Valore di ritorno

Nessuno.

TimeframesFlags

Il metodo restituisce il flag che indica i timeframes con una nuova barra.

```
int  TimeframesFlags(
   MqlDateTime&    time         // variabile per il tempo
   )
```

Parametri

time

[in] Variabile di tipo MqlDateTime per il nuovo orario, passata per riferimento.

Valore di ritorno

Esso restituisce la flag, che indica timeframes con una nuova barra.

CExpertSignal

CExpertSignal è una classe base per i segnali di trading, non fa nulla (ad eccezione dei metodi CheckReverseLong() e CheckReverseShort()), ma fornisce le interfacce.

Come usarla:

1. Prepara un algoritmo per i segnali di trading;
2. Creare la propria classe di segnale di trading, ereditata dalla classe CExpertSignal;
3. Sovrascrivere i metodi virtuali nella tua classe con i propri algoritmi.

È possibile trovare un esempio di classi di segnale di trading nella cartella Expert\Signal\.

Descrizione

CExpertSignal è una classe base per l'implementazione di algoritmi di segnali di trading.

Dichiarazione

```
class CExpertSignal : public CExpertBase
```

Titolo

```
#include <Expert\ExpertSignal.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertSignal

Discendenti diretti

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

Metodi della Classe

Inizializzazione	
virtual InitIndicators	Inizializza indicatori e serie temporali
virtual ValidationSettings	Controlla le impostazioni
virtual AddFilter	Aggiunge un filtro per segnale combinato
L'accesso ai Dati Protetti	
BasePrice	Imposta il livello di prezzo base
UsedSeries	Ottiene le flag di timeseries utilizzata
Impostazione Parametri	

Inizializzazione	
Weight	Imposta il valore del parametro "peso"("Weight")
PatternsUsage	Imposta il valore del parametro "PatternsUsage"
General	Imposta il valore del parametro "General"
Ignore	Imposta il valore del parametro "Ignore"
Invert	Imposta il valore del parametro "Invert"
ThresholdOpen	Imposta il valore del parametro "ThresholdOpen"
ThresholdClose	Imposta il valore del parametro "ThresholdClose"
PriceLevel	Imposta il valore del parametro "PriceLevel"
StopLevel	Imposta il valore del parametro "StopLevel"
TakeLevel	Imposta il valore del parametro "TakeLevel"
Expiration	Imposta il valore del parametro "Expiration"
Magic	Imposta il valore del parametro "Magic"
Controllo Condizioni di Trading	
virtual CheckOpenLong	Controlla condizioni per aprire la posizioni long
virtual CheckCloseLong	Controlla condizioni per chiudere la posizione long
virtual CheckOpenShort	Controlla condizioni per aprire posizioni short
virtual CheckCloseShort	Controlli condizioni per chiudere la posizione short
virtual CheckReverseLong	Controlla condizioni inversione di posizione long
virtual CheckReverseShort	Controlla condizioni inversione di posizione short
Impostazione Parametri di Trade	
virtual OpenLongParams	Imposta i parametri per l'apertura della posizione long
virtual OpenShortParams	Imposta i parametri per l'apertura della posizione short
virtual CloseLongParams	Imposta i parametri per la chiusura della posizione long
virtual CloseShortParams	Imposta i parametri per la chiusura della posizione short
Checking of Order Trailing Conditions	
virtual CheckTrailingOrderLong	Controlla condizioni per modificare i parametri dell'ordine pendente Buy
virtual CheckTrailingOrderShort	Controlla condizioni per modificare i parametri dell'ordine pendente Sell

Inizializzazione	
Metodi per controllare la formazione di Ordini di Mercato	
virtual LongCondition	Ottiene il risultato della verifica delle condizioni di buy
virtual ShortCondition	Ottiene il risultato della verifica delle condizioni di sell
virtual Direction	Ottiene la direzione "ponderata" del prezzo

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

BasePrice

Imposta il livello di prezzo base.

```
void  BasePrice(
   double    value         // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore del prezzo livello Base.

Valore di ritorno

Nessuno.

UsedSeries

Ottiene le flag delle timeseries utilizzate.

```
int  UsedSeries()
```

Valore di ritorno

Flag delle timeseries utilizzate (se il simbolo/ timeframe corrisponde simbolo/timeframe lavoranti), altrimenti 0.

Weight

Imposta nuovo valore del parametro "Weight".

```
void  Weight(
   double    value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "Weight".

Valore di ritorno

Nessuno.

PatternUsage

Imposta nuovo valore del parametro "PatternsUsage".

```
void  PatternUsage(
   double    value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "PatternsUsage".

Valore di ritorno

Nessuno.

General

Imposta nuovo valore del parametro "General".

```
void  General(
   int     value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "General".

Valore di ritorno

Nessuno.

Ignore

Imposta nuovo valore del parametro "Ignore".

```
void  Ignore(
   long     value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "Ignore".

Valore di ritorno

Nessuno.

Invert

Imposta nuovo valore del parametro "Invert".

```
void  Invert(
   long     value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "Invert".

Valore di ritorno

Nessuno.

ThresholdOpen

Imposta nuovo valore del parametro "ThresholdOpen".

```
void  ThresholdOpen(
   long     value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "ThresholdOpen".

Valore di ritorno

Nessuno.

Nota

L'intervallo del parametro "ThresholdOpen" è da 0 a 100. Utilizzato quando si "vota" per aprire la posizione.

ThresholdClose

Imposta nuovo valore del parametro "ThresholdClose".

```
void  ThresholdOpen(
   long     value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "ThresholdClose".

Valore di ritorno

Nessuno.

Nota

L'intervallo del parametro "ThresholdClose" è da 0 a 100. Utilizzato quando si "vota" per chiudere la posizione.

PriceLevel

Imposta nuovo valore del parametro "PriceLevel".

```
void  PriceLevel(
   double    value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore "PriceLevel".

Valore di ritorno

Nessuno.

Nota

Il valore di "PriceLevel" è definito in unità di livello dei prezzi. I valori numerici di unità di livello dei prezzi sono restituiti dal metodo PriceLevelUnit(). Il "PriceLevel" è utilizzato per definire il prezzo di apertura rispetto al prezzo base.

StopLevel

Imposta nuovo valore del parametro "StopLevel".

```
void  StopLevel(
   double    value         // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "StopLevel".

Valore di ritorno

Nessuno.

Nota

Il valore di "StopLevel" è definito in unità del livello dei prezzi. I valori numerici di unità di livello dei prezzi sono restituiti dal metodo PriceLevelUnit(). Lo "StopLevel" è utilizzato per definire il prezzo di Stop Loss rispetto al prezzo di apertura.

TakeLevel

Imposta nuovo valore del parametro "TakeLevel".

```
void  TakeLevel(
   double    value         // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "TakeLevel".

Valore di ritorno

Nessuno.

Nota

Il valore di "TakeLevel" è definito in unità di livello dei prezzi. I valori numerici di unità di livello dei prezzi sono restituiti dal metodo PriceLevelUnit(). Lo "TakeLevel" è utilizzato per definire il prezzo di Take Profit rispetto al prezzo di apertura.

Expiration

Imposta il valore del parametro "Espirazione".

```
void  Expiration(
   int     value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "Expiration".

Valore di ritorno

Nessuno.

Nota

Il valore del parametro "Expiration" è definito in barre. È usato come tempo di scadenza per Ordini Pendenti (quando il trading usa ordini pendenti).

Magic

Imposta il valore del parametro "Magic".

```
void  Magic(
   int     value          // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore di "Magic" (Expert Advisor ID).

Valore di ritorno

Nessuno.

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

InitIndicators

Inizializza tutti gli indicatori e le timeseries.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // puntatore
   )
```

Parametri

indicators

[in] Puntatore alla raccolta di indicatori e timeseries.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

I timeseries vengono inizializzati solo se l'oggetto utilizza il simbolo o il timeframe, diversi dal simbolo o timeframe definiti in fase di inizializzazione.

AddFilter

Aggiunge un filtro per il segnale composito.

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // puntatore
   )
```

Parametri

indicators

[in] Puntatore all'oggetto filtro.

Valore di ritorno

true in caso di successo, altrimenti false.

CheckOpenLong

Controlla condizioni per aprire la posizioni long

```
virtual bool  CheckOpenLong(
   double&    price,          // prezzo
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // scadenza
   )
```

Parametri

price

[in] [out] variabile per prezzo, passato per riferimento.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

expiration

[in][out] Variabile per l'orario d'espirazione, passato per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

CheckOpenShort

Controlla condizioni per aprire posizioni short

```
virtual bool  CheckOpenShort(
   double&    price,          // prezzo
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // scadenza
   )
```

Parametri

price

[in] [out] variabile per prezzo, passato per riferimento.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

expiration

[in][out] Variabile per l'orario d'espirazione, passato per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

OpenLongParams

Consente di impostare i parametri per aprire posizioni long.

```
virtual bool  OpenLongParams(
   double&    price,          // prezzo
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // scadenza
   )
```

Parametri

price

[in] [out] variabile per prezzo, passato per riferimento.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

expiration

[in][out] Variabile per l'orario d'espirazione, passato per riferimento.

Valore di ritorno

true in caso di successo, altrimenti false.

OpenShortParams

Consente di impostare i parametri per aprire posizioni short.

```
virtual bool  OpenShortParams(
   double&    price,          // prezzo
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // scadenza
   )
```

Parametri

price

[in] [out] variabile per prezzo, passato per riferimento.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

expiration

[in][out] Variabile per l'orario d'espirazione, passato per riferimento.

Valore di ritorno

true in caso di successo, altrimenti false.

CheckCloseLong

Controlla le condizioni per chiudere la posizione long.

```
virtual bool  CheckCloseLong(
   double&  price      // prezzo
   )
```

Parametri

price

[in][out] Variabile del prezzo close, passato per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

CheckCloseShort

Controlla le condizioni per chiudere la posizione short.

```
virtual bool  CheckCloseShort(
   double&  price      // prezzo
   )
```

Parametri

price

[in][out] Variabile del prezzo close, passato per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

CloseLongParams

Imposta i parametri per chiudere la posizione long.

```
virtual bool  CloseLongParams(
   double&    price          // prezzo
   )
```

Parametri

price

[in][out] Variabile del prezzo close, passato per riferimento.

Valore di ritorno

true in caso di successo, altrimenti false.

CloseShortParams

Imposta i parametri per chiudere la posizione short.

```
virtual bool  CloseShortParams(
   double&    price          // prezzo
   )
```

Parametri

price

[in][out] Variabile del prezzo close, passato per riferimento.

Valore di ritorno

true in caso di successo, altrimenti false.

CheckReverseLong

Controlla condizioni di inversione di posizione long.

```
virtual bool  CheckReverseLong(
   double&    price,          // prezzo
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // scadenza
   )
```

Parametri

price

[in] [out] variabile per prezzo, passato per riferimento.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

expiration

[in][out] Variabile per l'orario d'espirazione, passato per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

CheckReverseShort

Controlla condizioni di inversione di posizione short.

```
virtual bool  CheckReverseShort(
   double&    price,          // prezzo
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // scadenza
   )
```

Parametri

price

[in][out] Variable per prezzo di inversione, passate per riferimento.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

expiration

[in][out] Variabile per l'orario d'espirazione, passato per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

CheckTrailingOrderLong

Controlla condizioni per la modifica dei parametri di ordini pendenti Buy.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // ordine
   double&         price           // prezzo
   )
```

Parametri

order

[in] Puntatore alla classe oggetto COrderInfo .

price

[in][out] Variabile per prezzo Stop Loss.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

CheckTrailingOrderShort

Controlli condizioni per modifica dei parametri di ordini pendenti Sell.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // ordine
   double&         price           // prezzo
   )
```

Parametri

order

[in] Puntatore alla classe oggetto COrderInfo .

price

[in][out] Variabile per prezzo Stop Loss.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

LongCondition

Controlla condizioni per aprire la posizioni long

```
virtual int  LongCondition()
```

Valore di ritorno

Se le condizioni sono soddisfatte, restituisce il valore da 1 a 100 (a seconda della "forza" di un segnale). Se non c'è un segnale per aprire posizione long, restituisce 0.

Nota

Il metodo LongCondition() di una classe base non ha alcuna implementazione di verifica delle condizioni per aprire posizioni long e restituisce sempre 0.

ShortCondition

Controlla condizioni per aprire posizioni short

```
virtual int  ShortCondition()
```

Valore di ritorno

Se le condizioni sono soddisfatte, restituisce il valore da 1 a 100 (a seconda della "forza" di un segnale). Se non c'è un segnale per aprire la posizione short, restituisce 0.

Nota

Il metodo ShortCondition() di una classe base non ha alcuna implementazione di verifica delle condizioni per aprire posizioni short e restituisce sempre 0.

Direction

Restituisce il valore della direzione "ponderata"(weighted).

```
virtual double  Direction()
```

Valore di ritorno

Restituisce il valore>0 quando la direzione va verso l'alto (probabilmente) e il valore restituito <0 quando la direzione è verso il basso. Il valore assoluto dipende dalla "forza" (strenght) di un segnale.

Nota

Se si utilizzano i filtri, il risultato dipenderà dai filtri.

CExpertTrailing

CExpertTrailing è una classe base per gli algoritmi di trailing, non fa nulla, ma fornisce le interfacce.

Come usarla:

1. Prepara un algoritmo per il trailing;
2. Creare la propria classe di trailing, ereditata dalla classe CExpertTrailing;
3. Sovrascrivere i metodi virtuali nella tua classe con i propri algoritmi.

È possibile trovare un esempio di classi di trailing nella cartella Expert\Trailing\.

Descrizione

CExpertTrailing è una classe base per l'implementazione di algoritmi di trailing stop.

Dichiarazione

```
class CExpertTrailing : public CExpertBase
```

Titolo

```
#include <Expert\ExpertTrailing.mqh>
```

Gerarchia di ereditarietà

CObject

CExpertBase

CExpertTrailing

Discendenti diretti

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

Metodi della Classe

Controllo delle condizioni di Trailing Stop	
virtual CheckTrailingStopLong	Controlla condizioni per modificare i parametri della posizione long
virtual CheckTrailingStopShort	Controlla condizioni per modificare i parametri della posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

CheckTrailingStopLong

Controlla condizioni per modificare i parametri della posizione long.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // puntatore
   double&         sl,           // Stop Loss
   double&         tp            // Take Profit
   )
```

Parametri

position

[in] Puntatore alla classe oggetto CPositionInfo.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

Nota

Il metodo CheckTrailingStopLong(...) della classe base restituisce sempre false.

CheckTrailingStopShort

Controlla condizioni per modificare i parametri della posizione short.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // puntatore
   double&         sl,           // Stop Loss
   double&         tp            // Take Profit
   )
```

Parametri

position

[in] Puntatore alla classe oggetto CPositionInfo.

sl

[in][out] Variabile per il prezzo Stop Loss, passata per riferimento.

tp

[in][out] Variabile per il prezzo Take Profit, passata per riferimento.

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

Nota

Il metodo CheckTrailingStopShort(...) della classe base restituisce sempre false.

CExpertMoney

CExpertMoney è una classe base per gli algoritmi di gestione del denaro e del rischio.

Descrizione

CExpertMoney è una classe base per l'implementazione delle classi di gestione del denaro e del rischio.

Dichiarazione

```
class CExpertMoney : public CObject
```

Titolo

```
#include <Expert\ExpertMoney.mqh>
```

Gerarchia di ereditarietà

CObject

CExpertBase

CExpertMoney

Discendenti diretti

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

Metodi della Classe

L'accesso ai Dati Protetti	
Percent	Imposta il valore del parametro "Rischio percentuale" (Risk percent)
Inizializzazione	
virtual ValidationSettings	Controlla le impostazioni
Controllo delle Condizioni di Trading	
virtual CheckOpenLong	Ottiene il volume per la posizione long
virtual CheckOpenShort	Ottiene il volume per la posizione short
virtual CheckReverse	Ottiene il volume per l'inversione della posizione
virtual CheckClose	Controlli condizioni per chiudere la posizione aperta

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Percent

Imposta il valore del parametro "Rischio percentuale" (Risk percent).

```
void  Percent(
   double  percent      // rischio percentuale
   )
```

Parametri

percent

[in] Rischio percentuale.

Valore di ritorno

Nessuno.

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Il metodo ValidationSettings() della classe base restituisce sempre true.

CheckOpenLong

Ottiene il volume per la posizione long

```
virtual double  CheckOpenLong(
   double  price,     // prezzo
   double  sl         // Stop Loss
   )
```

Parametri

price

[in] Prezzo di Apertura della posizione long.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trading per la posizione long.

CheckOpenShort

Ottiene il volume per la posizione short.

```
virtual double  CheckOpenShort(
   double  price,     // prezzo
   double  sl         // Stop Loss
   )
```

Parametri

price

[in] Prezzo di apertuare per la posizione short.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trade per la posizione short.

CheckReverse

Ottiene il volume per l'inversione della posizione.

```
virtual double  CheckReverse(
   CPositionInfo*  position,     // puntatore
   double          sl            // Stop Loss
   )
```

Parametri

position

[in] Pointer to CPositionInfo class object.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

Volume per l'inversione della posizione.

CheckClose

Controlli condizioni per chiudere la posizione aperta.

```
virtual double  CheckClose()
```

Valore di ritorno

true se la condizione è soddisfatta, altrimenti false.

I moduli di Segnali di Trade

La consegna standard del terminale client include una serie di moduli di segnali di trade giai pronti per l' "MQL5 Wizard". Durante la creazione guidata di un Expert Advisor con MQL5 Wizard, è possibile utilizzare qualsiasi combinazione di moduli di segnali di trade (fino a 64). La decisione finale relativa all'operazione di trade viene effettuata sulla base di un'analisi complessa di segnali ottenuti da tutti i moduli inclusi. La descrizione dettagliata del meccanismo di presa di decisioni di trade è data sotto.

La fornitura standard comprende i seguenti moduli di segnali:

- Segnali del indicatore Accelerator Oscillator
- I segnali dell'indicatore Adaptive Moving Average
- I Segnali dell' Indicatore Awesome Oscillator
- Segnali di Oscillator Bears Power
- Segnali dell'oscillatore Power Bulls
- Segnali dell'Indice Oscillator Commodity Channel Index
- Segnali del DeMarker Oscillator
- I segnali dell' Indicatore Double Exponential Moving Average
- Segnali del indicatore Indicator Envelopes
- I segnali dell'indicatore Fractal Adaptive Moving Average
- Segnali del Filtro Orario Intraday
- I segnali del MACD Oscillator
- I segnali dell'indicatore Moving Average
- I segnali dell'indicatore Parabolic SAR
- I segnali dell'oscillatore Relative Strength Index
- I segnali dell'oscillatore Relative Vigor Index
- I segnali dell'Oscillatore Stochastic
- I segnali dell'oscillatore Triple Exponential Average
- I segnali dell' Indicatore Triple Exponential Moving Average
- Segnali dell'oscillatore Williams Percent Range

Il meccanismo di presa di decisioni di Trade sulla base di Moduli di Segnale

Il meccanismo di presa decisioni di trade può essere rappresentato come il seguente elenco dei principi fondamentali:

- Ciascuno dei moduli di segnali ha il suo insieme di moduli di mercato (una certa combinazione di prezzi ed i valori di un indicatore).
- Ciascun modello di mercato ha un significato che può variare con la gamma da 1 a 100. Più alto è il significato, più forte è il modello.
- Ciascuno dei modelli genera una previsione della direzione del movimento di prezzo.
- Una previsione di un modulo è il risultato della ricerca di modelli incorporati, ed è emesso come un numero compreso tra -100 e 100. Il segno determina la direzione del movimento previsto (segno

negativo significa che il prezzo scenderà, segno positivo significa che il prezzo salirà). Il valore assoluto corrisponde alla forza dei modelli migliori trovati.

- La previsione di ogni modulo è inviata al "voto" finale con un coefficiente di peso da 0 ad 1 specificato nelle impostazioni ("Weight").
- Il risultato del voto è un numero compreso tra -100 e 100, dove il segno determina la direzione del movimento previsto ed il valore assoluto caratterizza la forza del segnale. Viene calcolato come la media Aritmetica delle previsioni ponderate di tutti i moduli di segnali.

Ogni Expert Advisor generato ha due impostazioni regolabili — i livelli di soglia(threshold) di apertura e chiusura di una posizione (ThresholdOpen e ThresholdClose) che possono essere pari ad un valore nell'intervallo da 0 a 100. Se l'intensità del segnale finale supera un livello di soglia, viene eseguita un'operazione di trade che corrisponde al segno del segnale.

Esempi

Si consideri un Expert Advisor con le seguenti soglie: ThresholdOpen=20 e ThresholdClose=90. I due moduli di segnali partecipano alla presa di decisioni sulle operazioni di trade: il modulo MA con peso 0.4 ed il modulo Stochastic con peso 0.8. Analizziamo due varianti di segnali di trade ottenuti:

Variante 1.

Il prezzo ha attraversato il MA sorgente verso l'alto. Questo caso corrisponde ad uno dei modelli di mercato implementati nel modulo MA. Questo modello implica un aumento del prezzo. La sua importanze è uguale a 100. Allo stesso tempo, l'oscillatore Stochastic si abbassa e forma una divergenza con il prezzo. Questo caso corrisponde ad uno dei modelli implementati nel modulo Stochastic. Questo modello implica una diminuzione di prezzo. Il peso di questo modello è di 80.

Calcoliamo il risultato del "voto" finale. Il tasso ottenuto dal modulo MA è calcolato come 0,4 * 100 = 40. Il valore dal modulo Stochastic è calcolato come 0,8 * (-80) = -64. Il valore finale è calcolato come media aritmetica di queste due tariffe: (40 - 64)/2 =-12. Il risultato della votazione è il segnale per la vendita con forza relativa pari a 12. Il livello di soglia pari a 20 non viene raggiunto. Quindi l'operazione di trade non viene eseguita.

Variante 2.

Il prezzo ha attraversato il MA crescente verso il basso. Questo caso corrisponde ad uno dei modelli implementati nel modulo MA. Questo modello implica un aumento del prezzo. Il suo significato è uguale a 10. Allo stesso tempo, l'oscillatore Stochastic si abbassa e forma una divergenza con il prezzo. Questo caso corrisponde ad uno dei modelli implementati nel modulo Stochastic. Questo modello implica una diminuzione di prezzo. Il peso di questo modello è di 80.

Calcoliamo il risultato del "voto" finale. Il tasso ottenuto dal modulo MA è calcolato come 0,4 * 10 = 4. Il valore dal modulo Stochastic è calcolato come 0,8 * (-80) = -64. Il valore finale è calcolato come la media aritmetica di questi due valori: a (4 - 64)/2 = -30. Il risultato della votazione è il segnale per la vendita con forza relativa pari a 30. Il livello di soglia pari a 20 viene raggiunto. Così il risultato è il segnale per aprire una posizione short.

a) Divergenza del prezzo e l'oscillatore Stochastic (varianti 1 e 2).

b) il prezzo incrocia l'indicatore MA verso l'alto (variante 1).

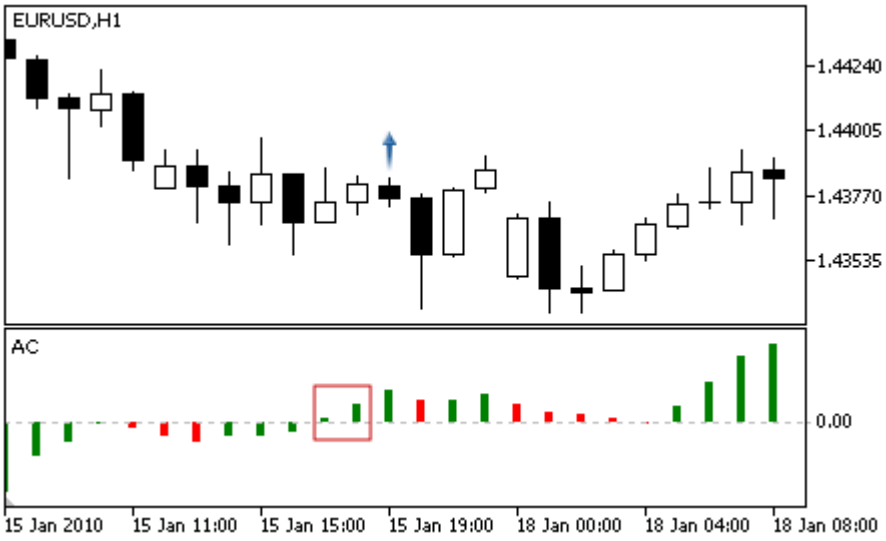
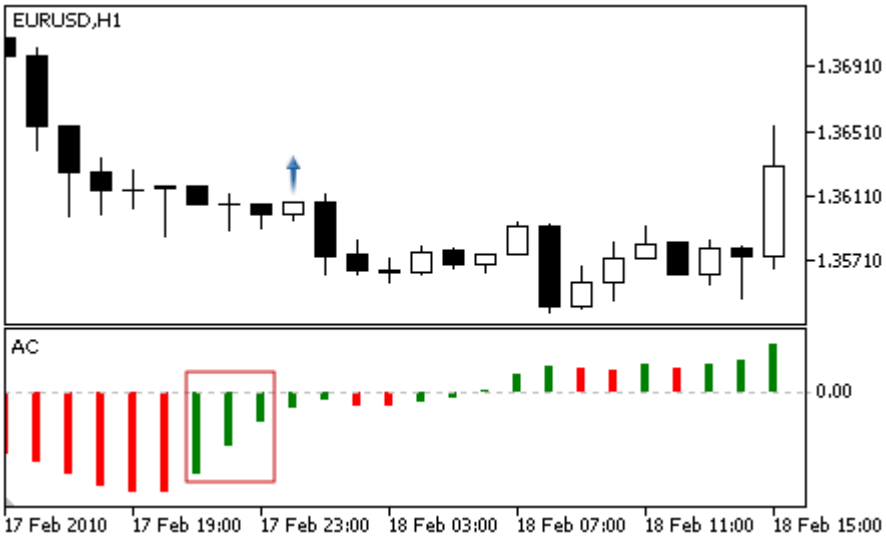
c) il prezzo incrocia l'indicatore MA verso il basso (varianti 2).

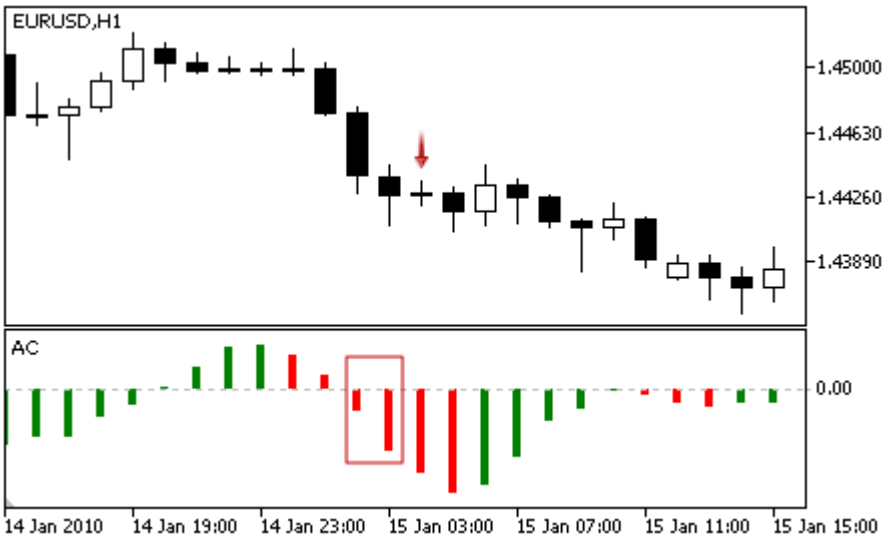
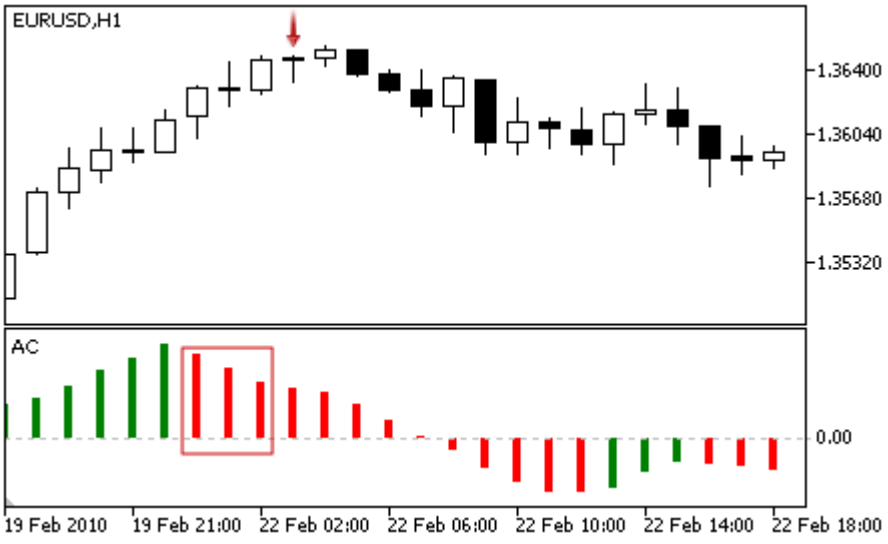
Segnali del indicatore Accelerator Oscillator

Questo modulo è basato sui modelli di mercato dell'indicatore Accelerator Oscillator. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• Il valore dell'indicatore è superiore a 0 e sorge alla barra analizzata e precedenti.  • Il valore dell'indicatore è inferiore a 0 e sorge alla barra analizzata e precedenti. 
Per il selling	• Il valore dell'indicatore è inferiore a 0 e cade alla barra analizzata e precedenti.

Tipo di segnale	Descrizione delle condizioni
	 • Il valore dell'indicatore è inferiore a 0 e cade alla barra analizzata e precedenti. 
Nessuna obiezione per il buying	Il valore dell'indicatore cresce alla barra analizzata.
Nessuna obiezione alla vendita	Il valore dell'indicatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

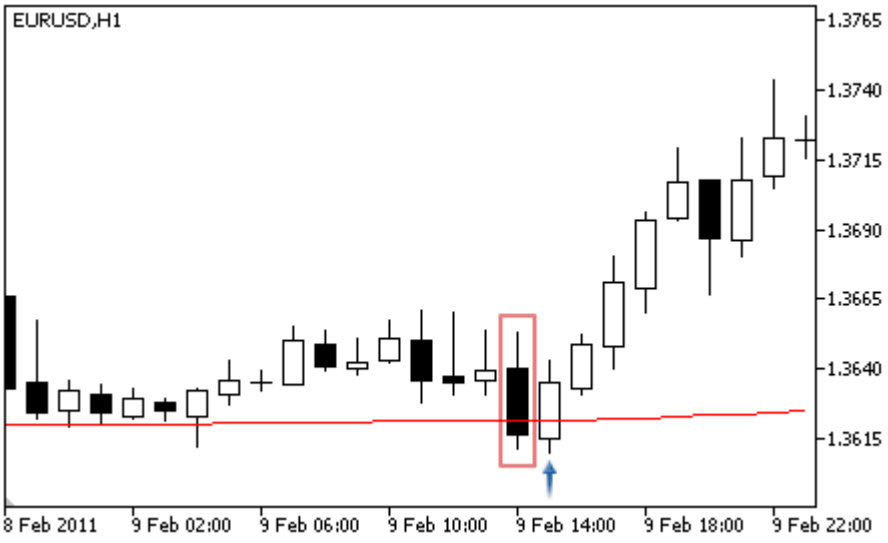
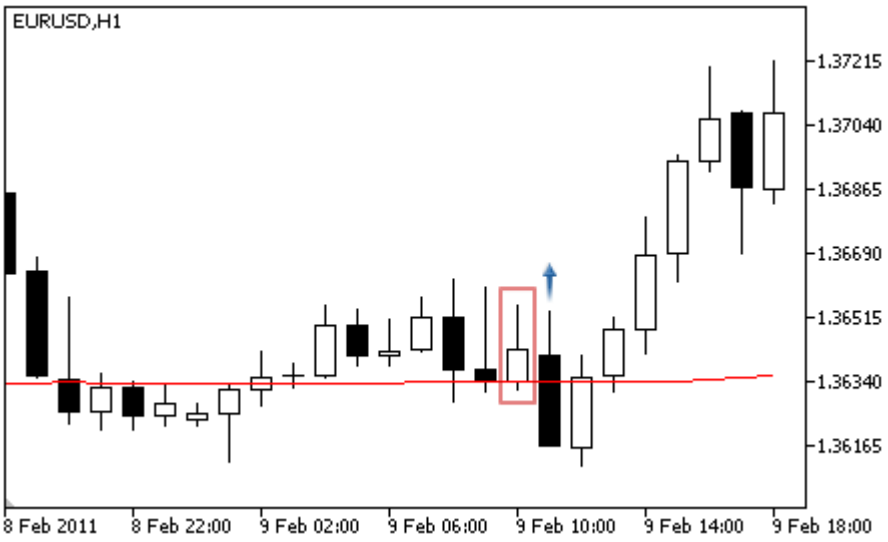
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.

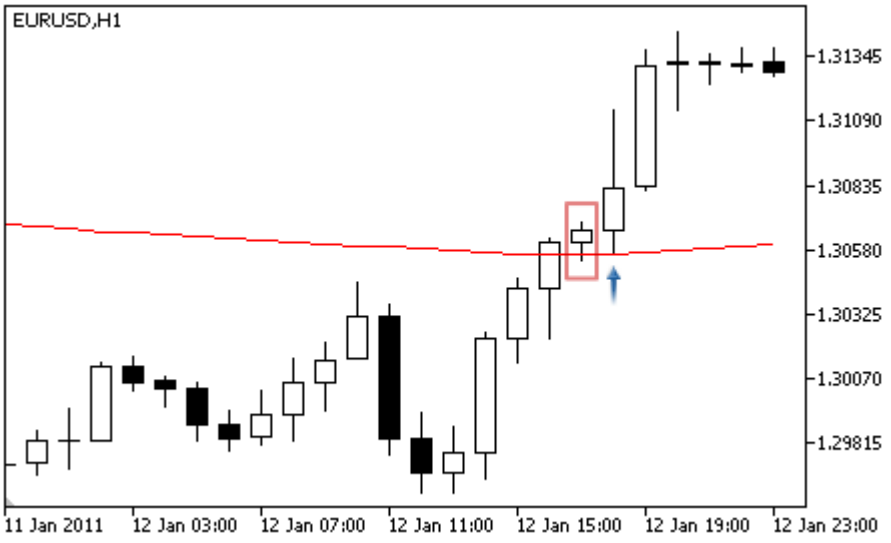
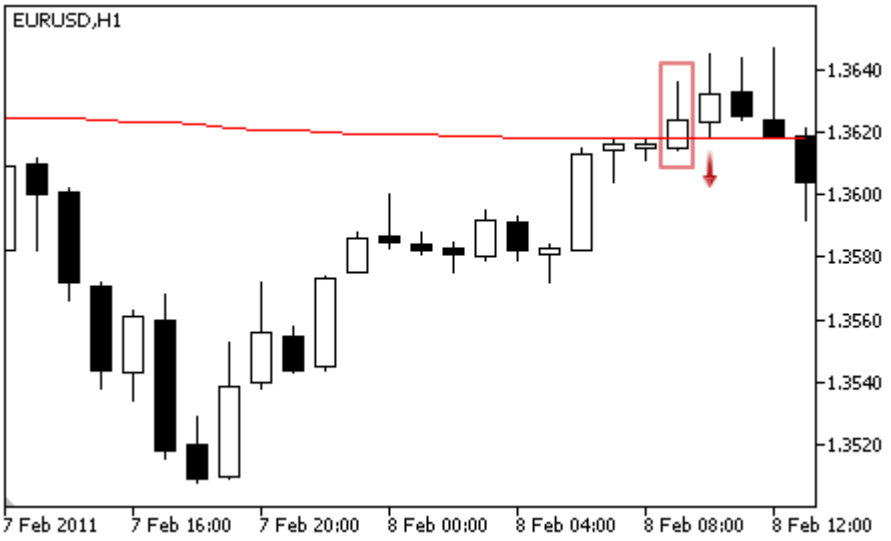
I segnali dell'indicatore Adaptive Moving Average

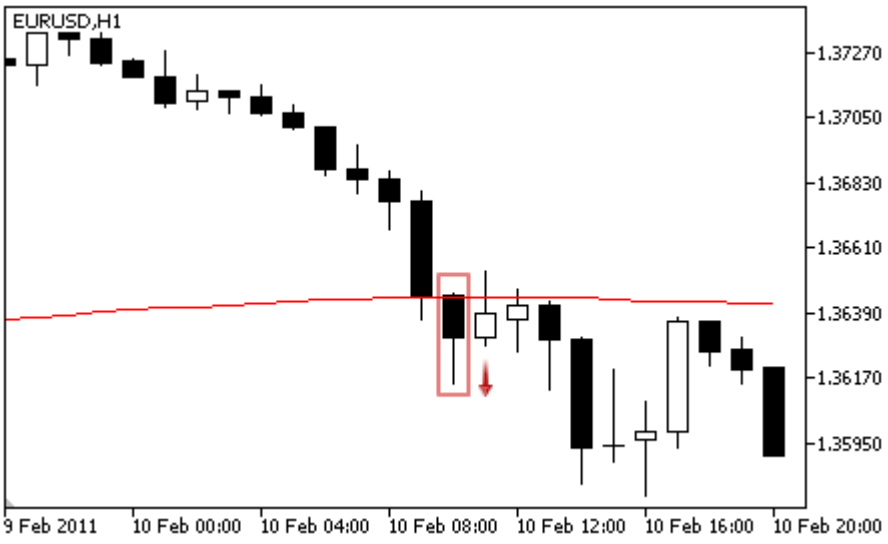
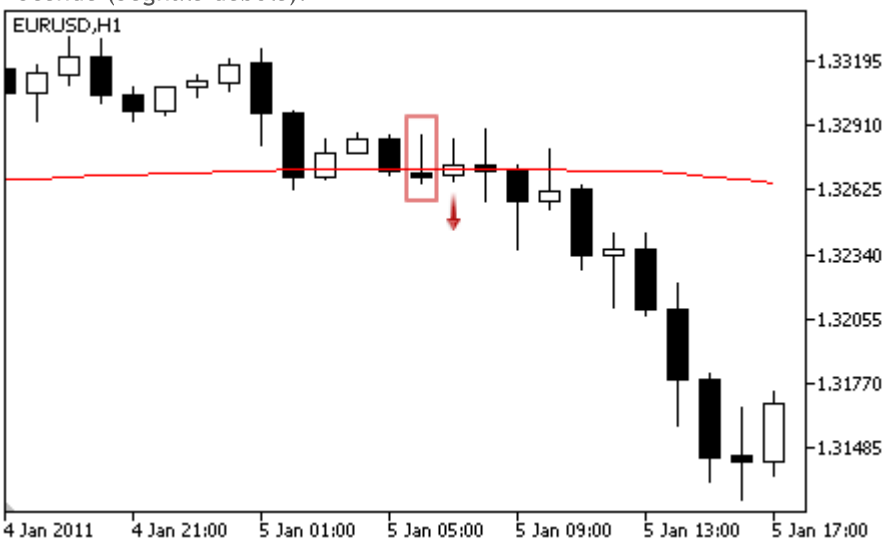
Questo modulo è basato sui modelli di mercato dell'indicatore Adaptive Moving Average. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• Il prezzo ha incrociato l'indicatore verso il basso (il prezzo di Apertura della barra analizzata è sopra l'indicatore e il prezzo di Chiusura è sotto l'indicatore) e l'indicatore sale (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di apertura della barra analizzata è sotto l'indicatore ed il prezzo di chiusura è sopra l'indicatore) e l'indicatore sale (segnale forte). 

Tipo di segnale	Descrizione delle condizioni
	• L'ombra inferiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono al di sopra dell'indicatore, ed il prezzo Basso(Low) è sotto l'indicatore) e l'indicatore aumenta (segnale debole). 
Per il selling	• Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di Apertura(Open) della barra analizzata è sotto l'indicatore ed il prezzo di Chiusura(Close) è sopra l' indicatore) e l'indicatore scende (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso il basso (il prezzo di apertura(Open) della barra analizzata è sopra l'indicatore ed il prezzo di chiusura(Close) è sotto l'indicatore) e l'indicatore scende (segnale forte).

Tipo di segnale	Descrizione delle condizioni
	 • L'ombra superiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono sotto l'indicatore, ed il prezzo High è al di sopra dell' indicatore) e l'indicatore scende (segnale debole). 
Nessuna obiezione per il buying	Il prezzo è sopra l'indicatore.
Nessuna obiezione alla vendita	Il prezzo è sotto l'indicatore.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

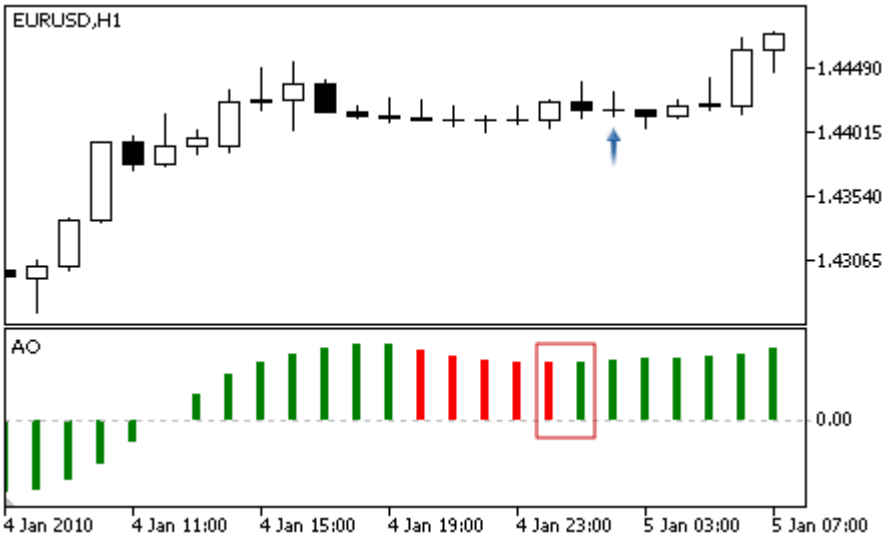
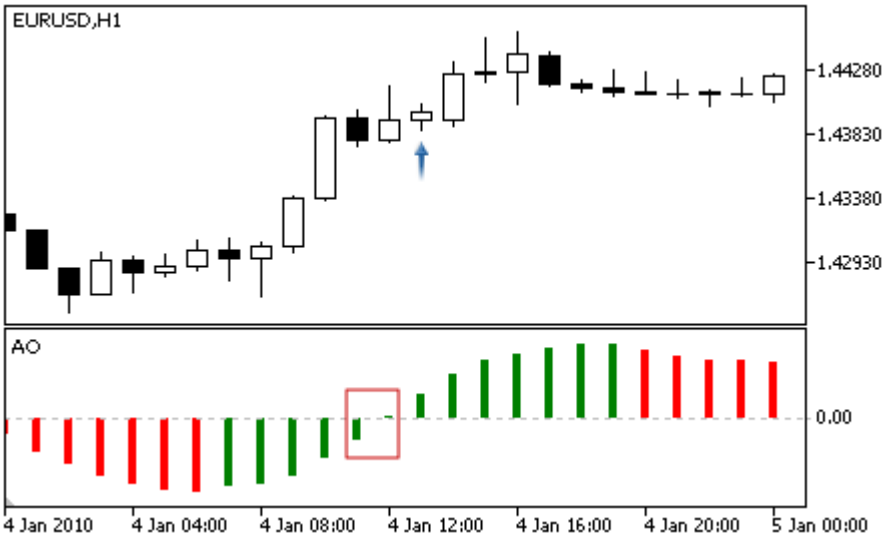
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodMA	Periodo di media dell'indicatore.
Shift	Slittamento dell'indicatore lungo l'asse temporale (in barre).
Metodo	Metodo di calcolo della media.
Applied	A price series usate per il calcolo dell'indicatore.

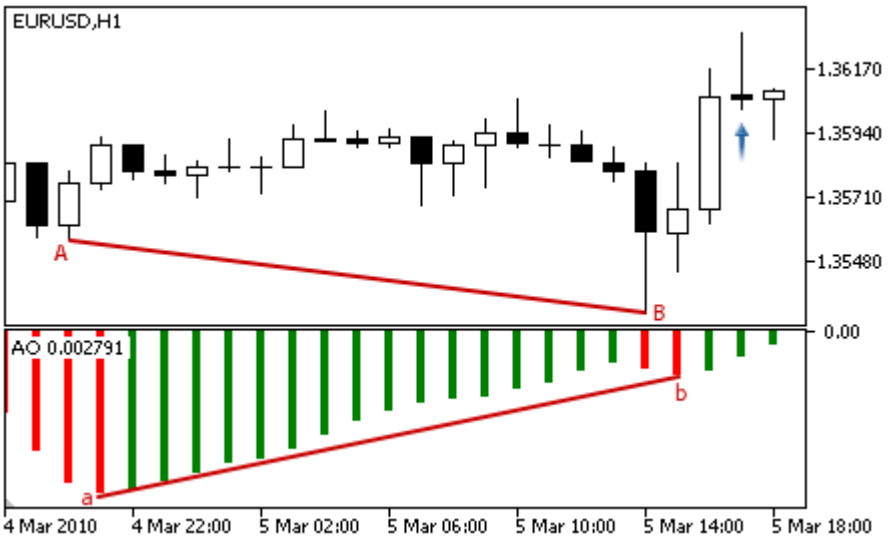
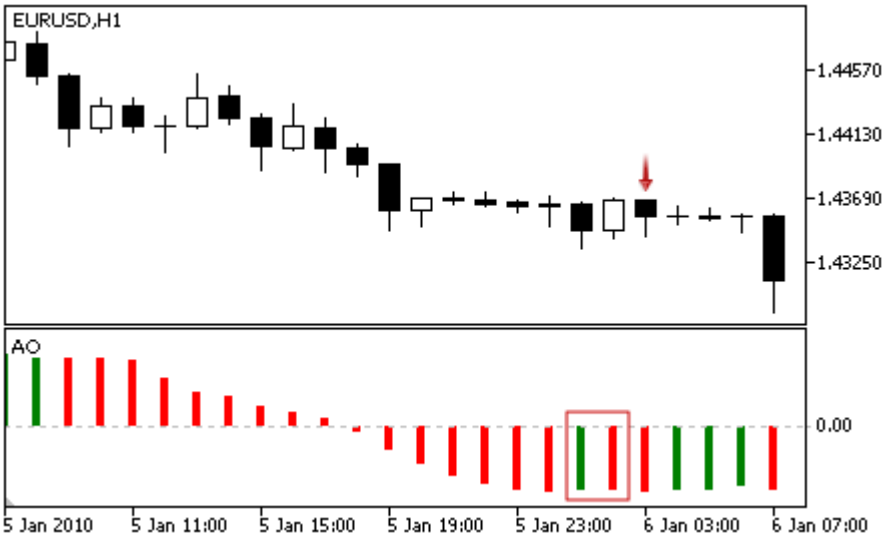
I Segnali dell' Indicatore Awesome Oscillator

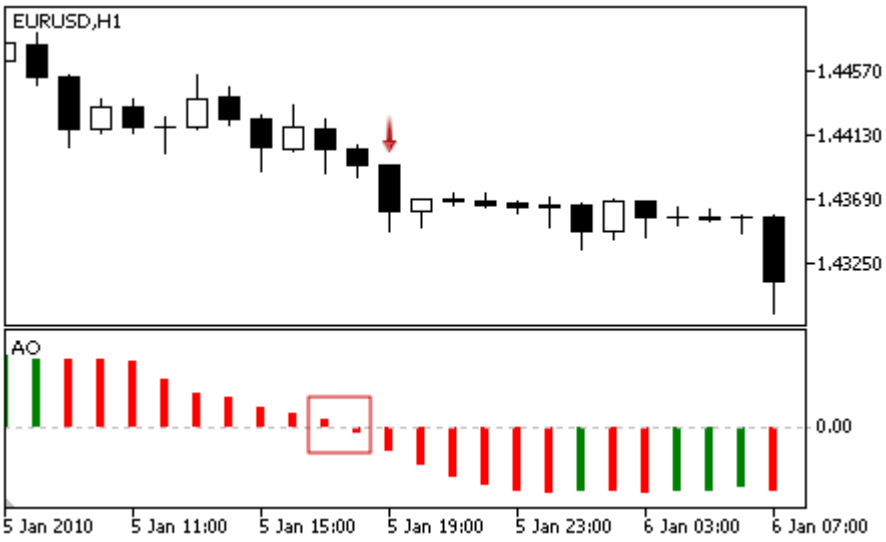
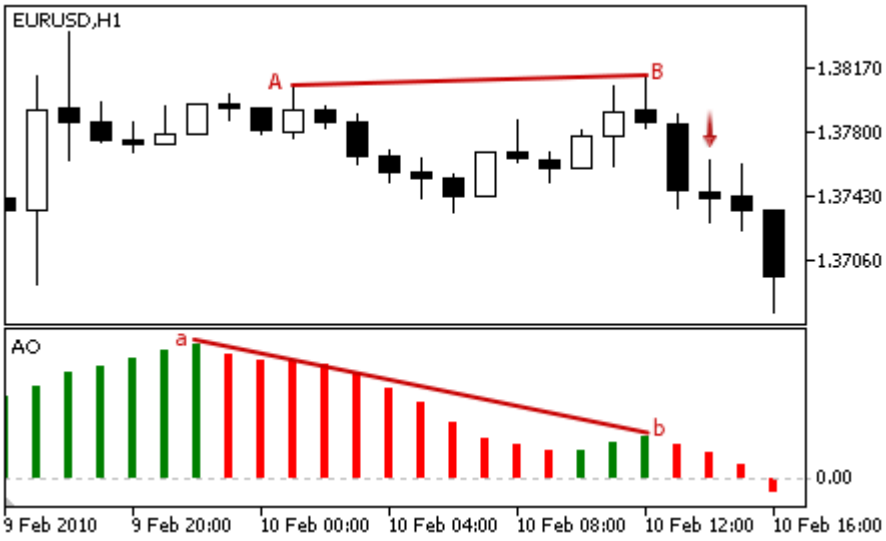
Questo modulo di segnali si basa sui modelli di mercato dell'indicatore Awesome Oscillator. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Saucer** — il valore dell'indicatore alla barra analizzata si alza, e cade alle barre precedenti; a ciò, entrambi i valori sono superiori 0.  • **Attraversando la linea dello zero** — il valore dell'indicatore è superiore a 0 alla barra analizzata, ed è inferiore a 0 alla barra precedente.  • **Divergenza** — Il primo fondo(parte bassa) analizzato dell'indicatore è meno profondo del precedente, e il corrispondente prezzo a valle è più

Tipo di segnale	Descrizione delle condizioni
	profondo di quello precedente. Inoltre, l'indicatore non deve superare il livello zero. 
Per il selling	• **Saucer** — il valore dell'indicatore alla barra analizzata cade, ed è salito alla barra precedente; al che, entrambi i valori sono inferiori a 0.  • **Incrociando la linea dello zero** — il valore dell'indicatore è inferiore a 0 alla barra analizzata, ed è superiore a 0 alla barra precedente.

Tipo di segnale	Descrizione delle condizioni
	 • **Divergenza** — il primo picco analizzato dell'indicatore è inferiore a quello precedente, ed il corrispondente picco prezzo è superiore a quello precedente. Inoltre, l'indicatore non deve scende sotto il livello zero. 
Nessuna obiezione per il buying	Il valore dell'indicatore cresce alla barra analizzata.
Nessuna obiezione alla vendita	Il valore dell'indicatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

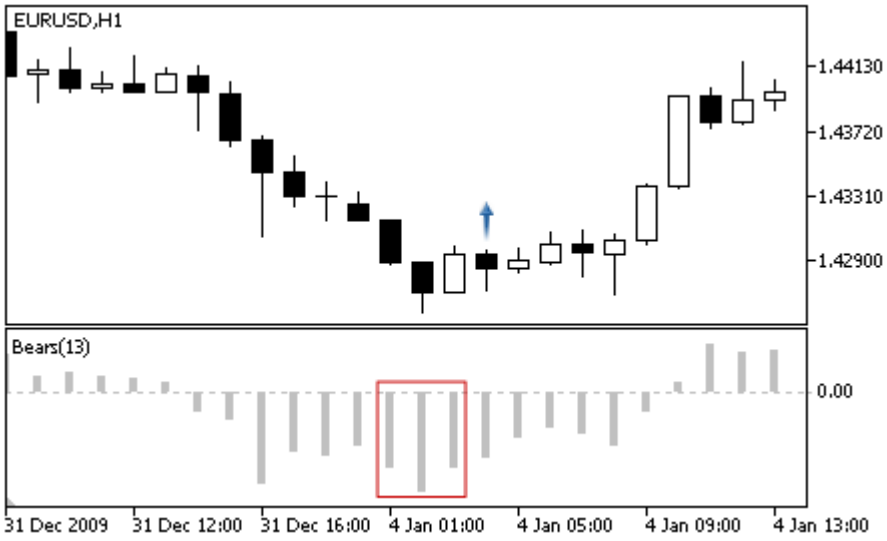
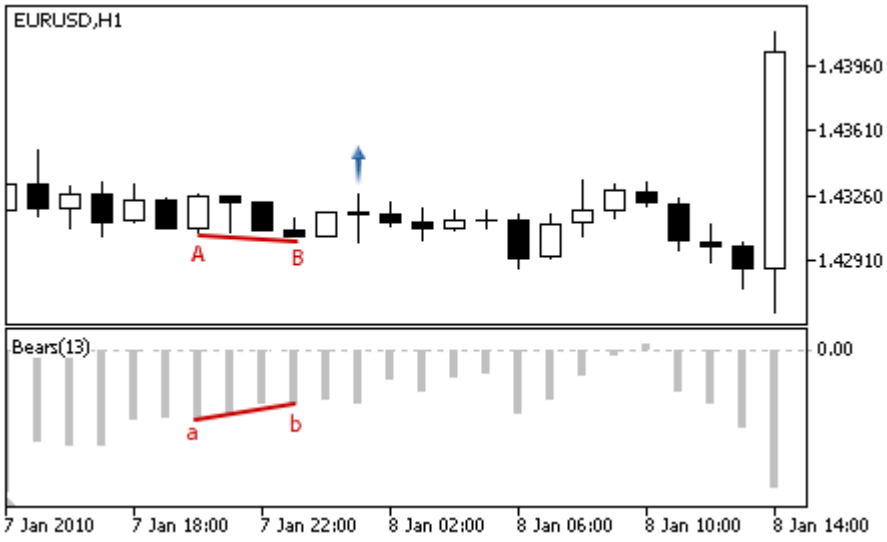
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.

Segnali di Oscillator Bears Power

Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore Bears Power. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Reverse** — l'oscillatore si rivolta verso l'alto ed il suo valore alla barra analizzata è inferiore a 0.  • **Divergenza** — Il primo fondo analizzato dell'oscillatore è superiore a quello precedente, ed il corrispondente prezzo di fondo è inferiore al precedente. Inoltre, l'oscillatore non deve superare il livello zero. 
Per il selling	Non ci sono segnali per il Selling.

Tipo di segnale	Descrizione delle condizioni
Nessuna obiezione per il buying	Il valore dell'oscillatore è minore di 0.
Nessuna obiezione alla vendita	Nessun segnale.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

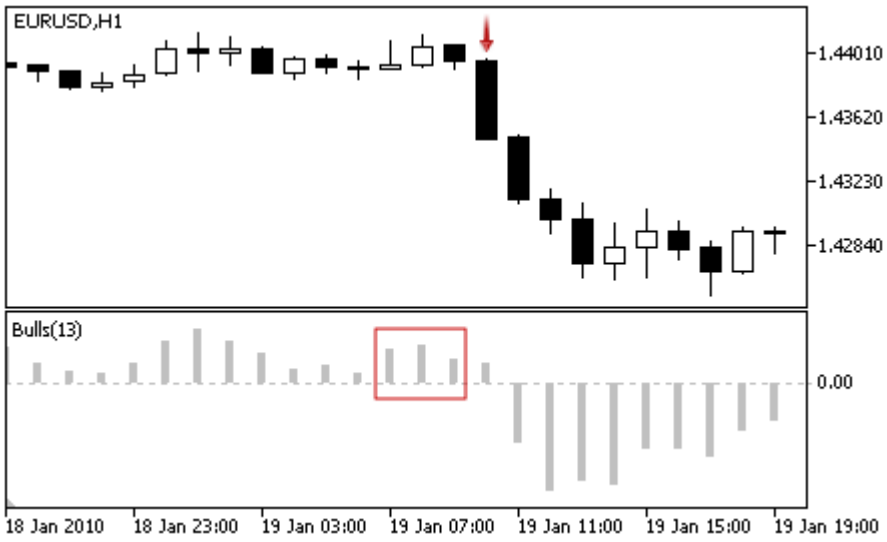
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodBears	Periodo di calcolo dell'oscillatore.

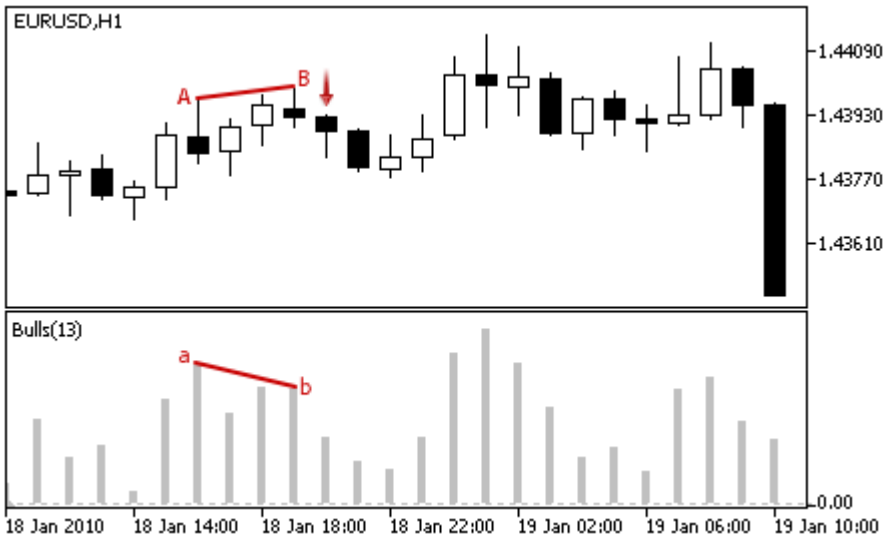
Segnali dell'oscillatore Power Bulls

Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore Bulls Power. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	Non ci sono segnali per il buying.
Per il selling	• **Reverse** — l'oscillatore si rivolta verso il basso e il suo valore alla barra analizzata è superiore a 0.  • **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente, ed il corrispondente prezzo picco è superiore al picco precedente. Inoltre, l'oscillatore non deve scendere al di sotto del livello zero.

Tipo di segnale	Descrizione delle condizioni
	
Nessuna obiezione per il buying	Nessun segnale.
Nessuna obiezione alla vendita	Il valore dell'oscillatore è maggiore di 0.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

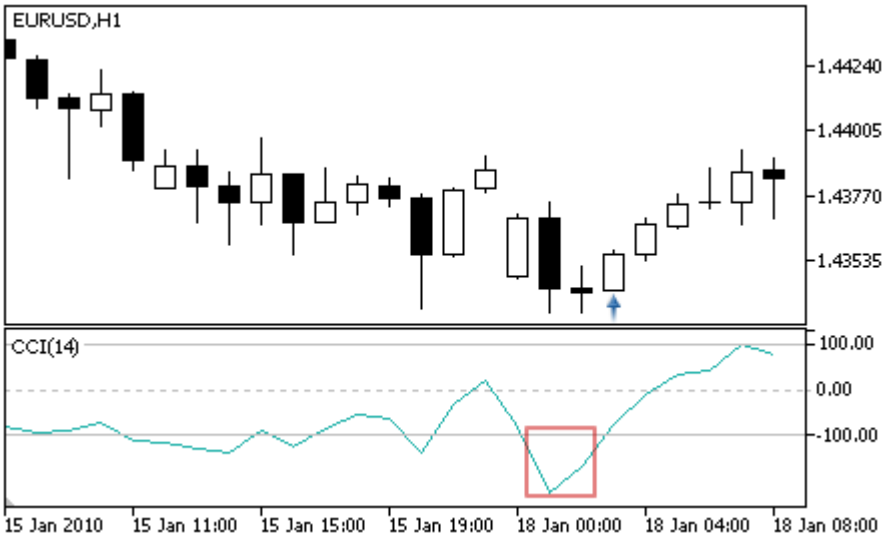
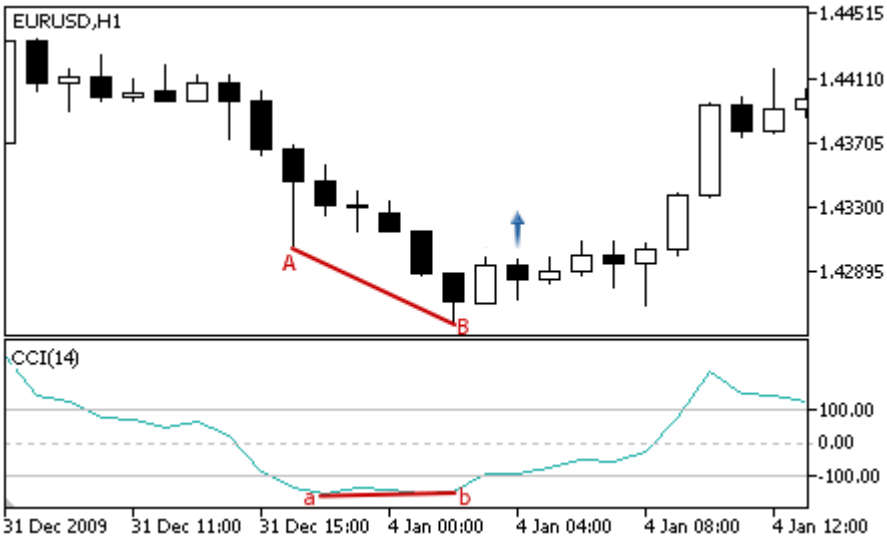
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodBulls	Periodo di calcolo dell'oscillatore.

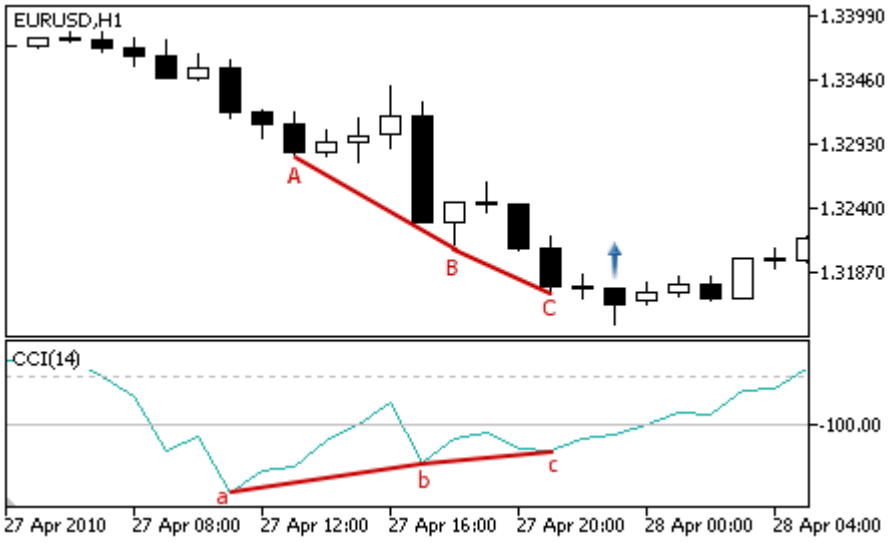
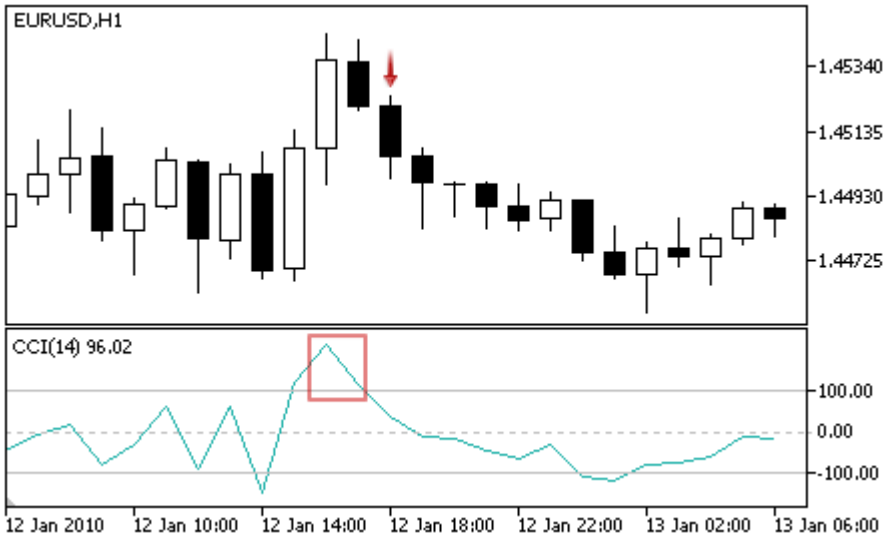
Segnali dell'Indice Oscillator Commodity Channel Index

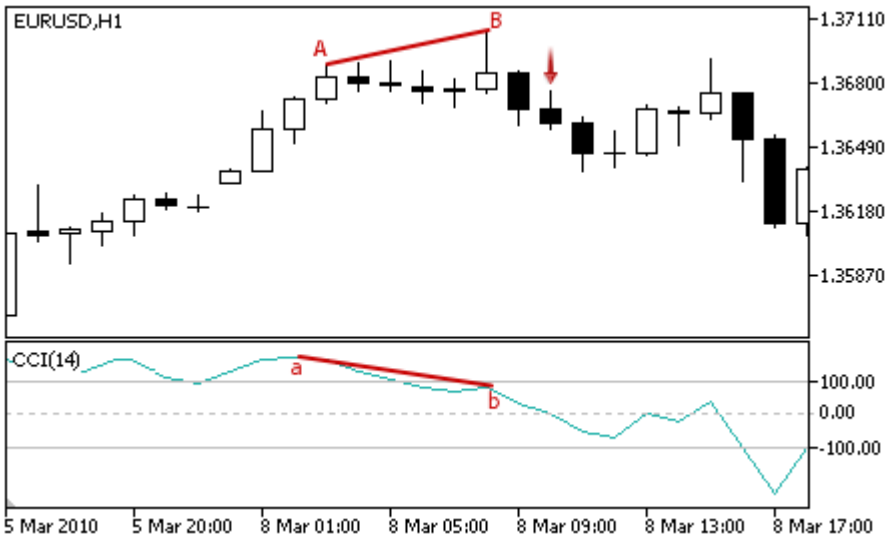
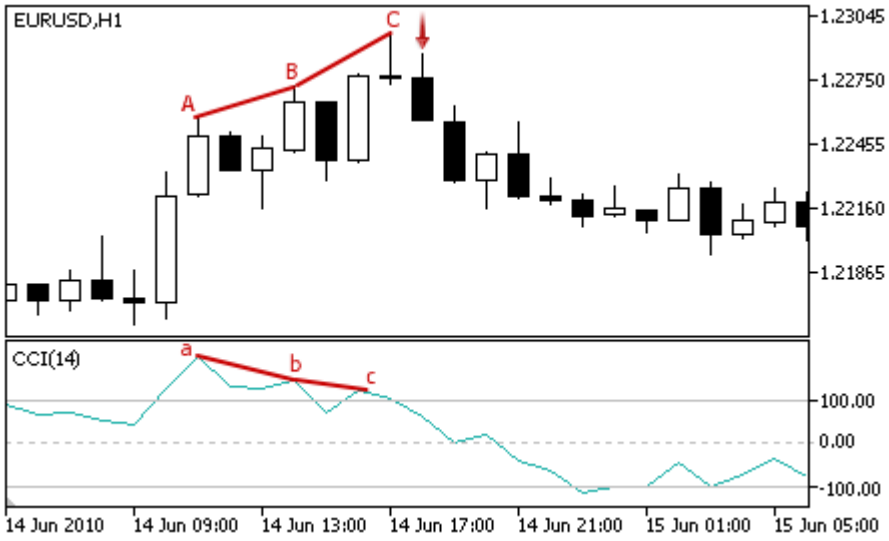
Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore Commodity Channel Index. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Reverse dietro il livello di sovravenduto(overselling)** — l'oscillatore si rivolta verso l'alto e il suo valore alla barra analizzata è dietro il livello di overselling (il valore di default è -100).  • **Divergenza** — il primo fondo analizzato dell'oscillatore è superiore a quello precedente, ed il corrispondente fondo(in basso) prezzo è inferiore al precedente. 

Tipo di segnale	Descrizione delle condizioni
	• **Doppia divergenza** — l'oscillatore forma tre conseguenti fondi, ciascuno di loro è superiore al precedente; ed il prezzo forma tre fondi corrispondenti, ciascuno di cui è inferiore a quello precedente. 
Per il selling	• **Reverse dietro il livello di ipercomprato(overbought)** — l'oscillatore si rivolta verso il basso e il suo valore alla barra analizzata è dietro il livello di ipercomprato (il valore di default è 100).  • **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente ed il prezzo picco corrispondente è superiore al picco precedente.

Tipo di segnale	Descrizione delle condizioni
	 • **Doppia divergenza** — l'oscillatore forma tre conseguenti picchi, ciascuno di loro è inferiore a quello precedente; il prezzo forma tre picchi corrispondenti, ciascuno dei quali è superiore a quello precedente. 
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

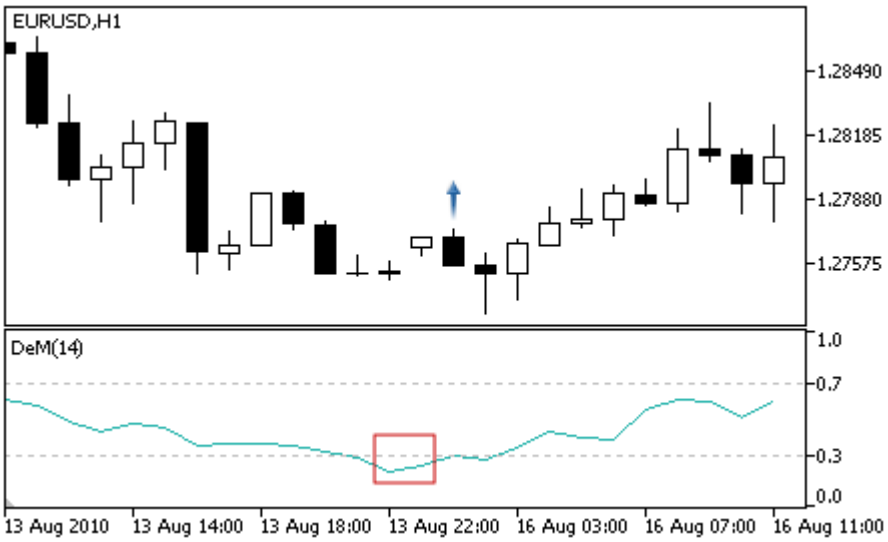
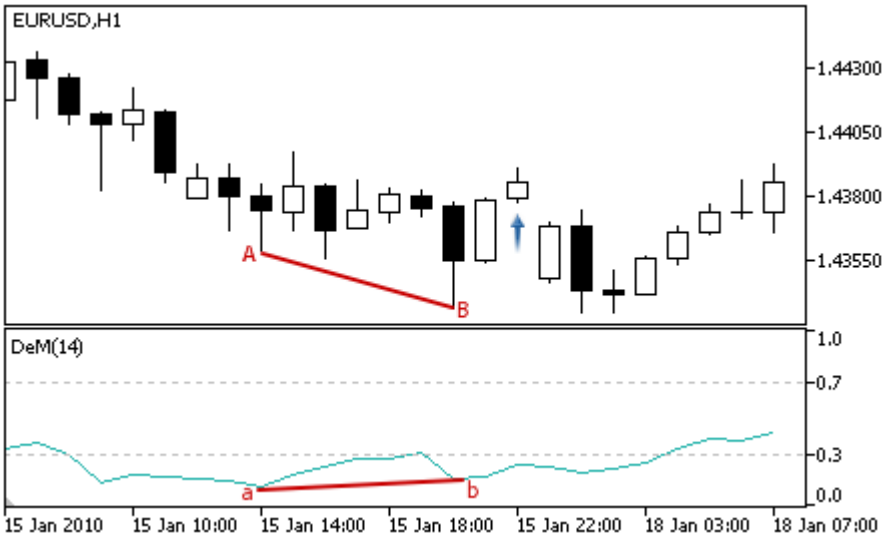
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodCCI	Periodo di calcolo dell'oscillatore.
Applied	Una serie di prezzi utilizzata per il calcolo dell'oscillatore.

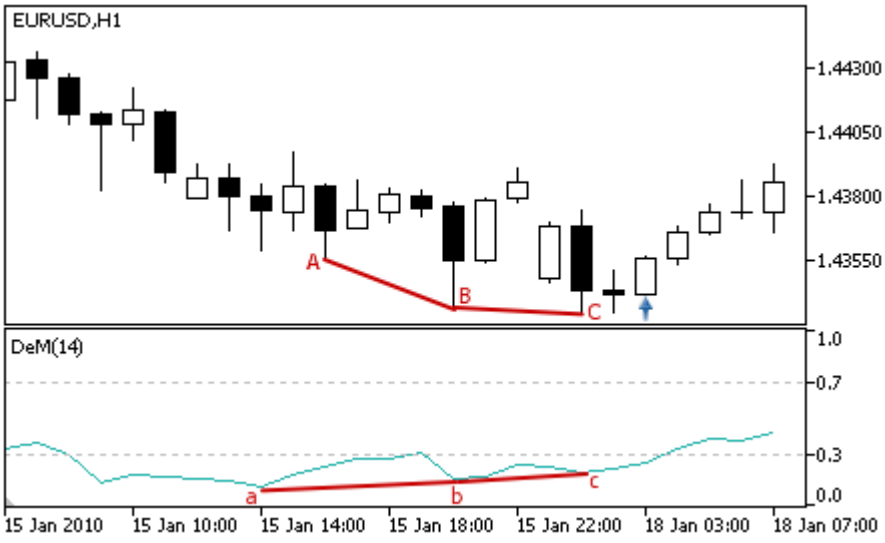
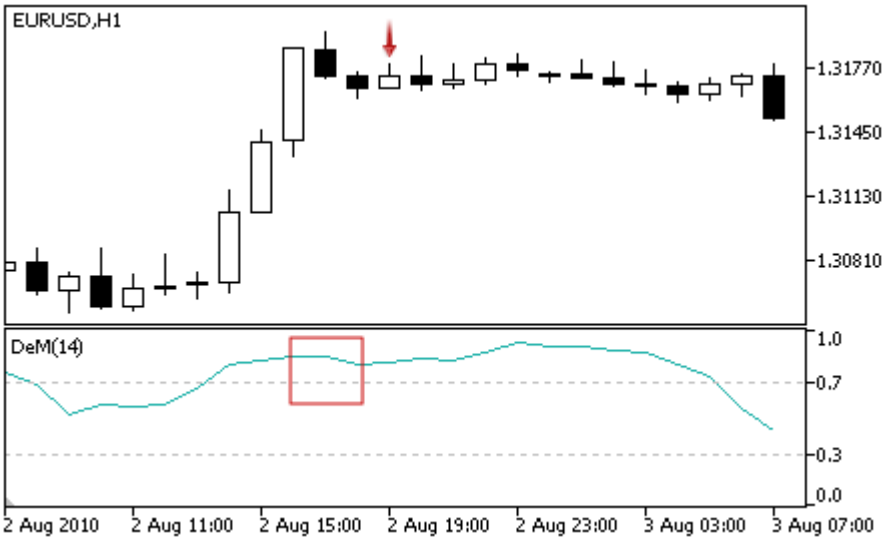
Segnali del DeMarker Oscillator

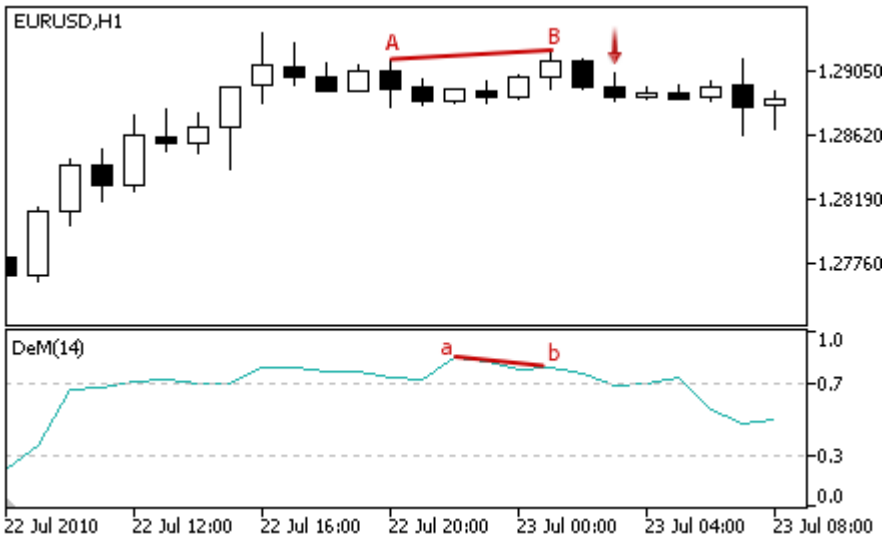
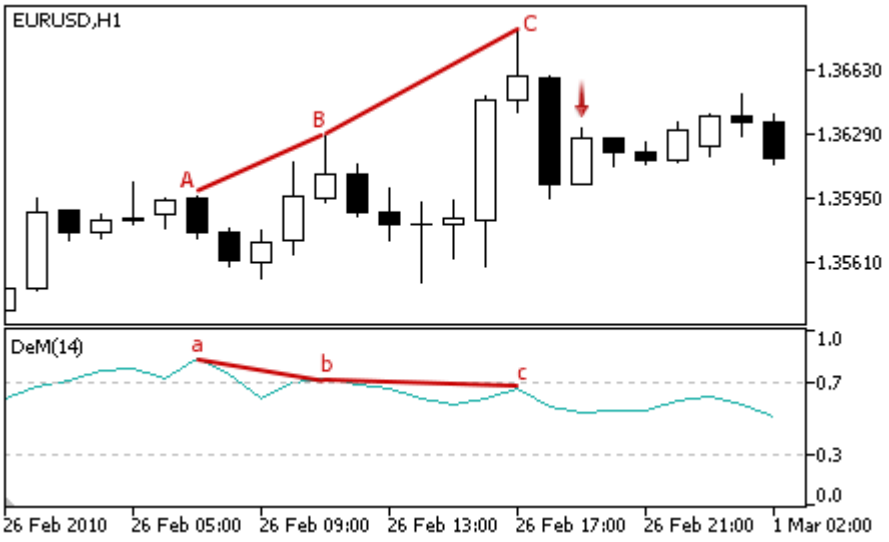
Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore DeMarker. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Inversione dietro il livello di ipervenduto(oversold)** - l'oscillatore si rivolta verso l'alto e il suo valore alla barra analizzata è dietro il livello di oversold (valore di default è 0.3).  • **Divergenza** — il primo fondo analizzato dell'oscillatore è superiore a quello precedente, ed il corrispondente fondo(in basso) prezzo è inferiore al precedente. 

Tipo di segnale	Descrizione delle condizioni
	• **Doppia divergenza(double divergence)** - l' oscillatore forma tre conseguenti fondi, ciascuno di loro è superiore al precedente; ed il prezzo ha formato tre fondi corrispondenti, e ciascuno di essi è inferiore a quello precedente. 
Per il selling	• **Inversione dietro il livello di ipercomprato** - l'oscillatore rivolto verso il basso e il suo valore al barra analizzata è dietro il livello di ipercomprato (valore di default è 0.7).  • **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente ed il prezzo picco corrispondente è superiore al picco precedente.

Tipo di segnale	Descrizione delle condizioni
	 • **Doppia divergenza** — l'oscillatore forma tre conseguenti picchi, ciascuno di loro è inferiore a quello precedente; il prezzo forma tre picchi corrispondenti, ciascuno dei quali è superiore a quello precedente. 
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

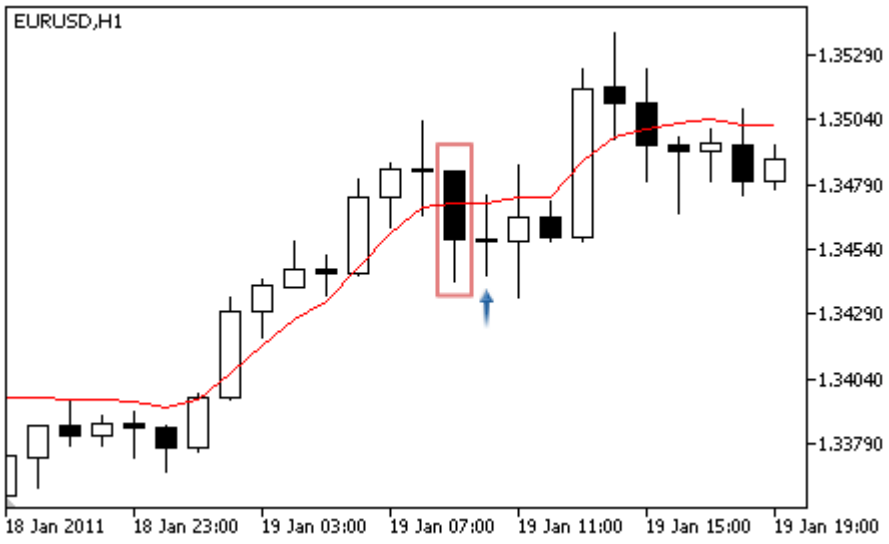
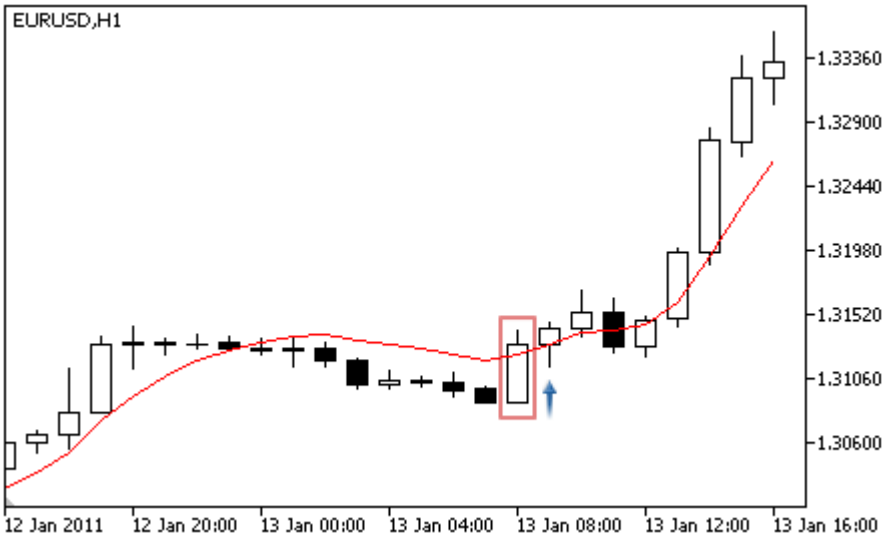
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodDeM	Periodo di calcolo dell'oscillatore.

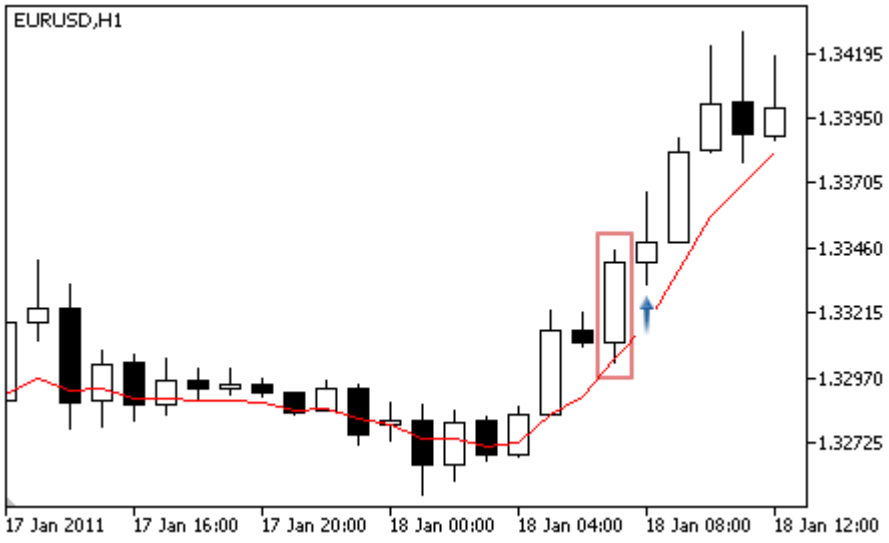
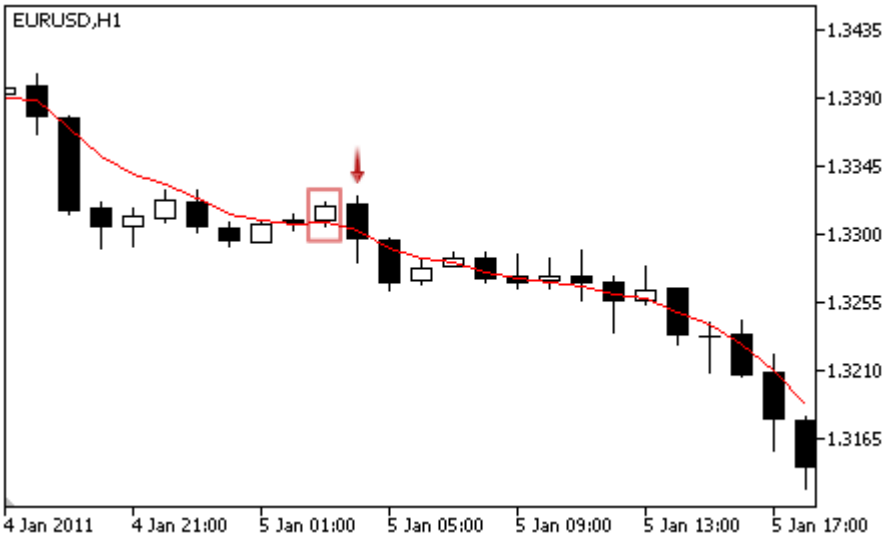
I segnali dell' Indicatore Double Exponential Moving Average

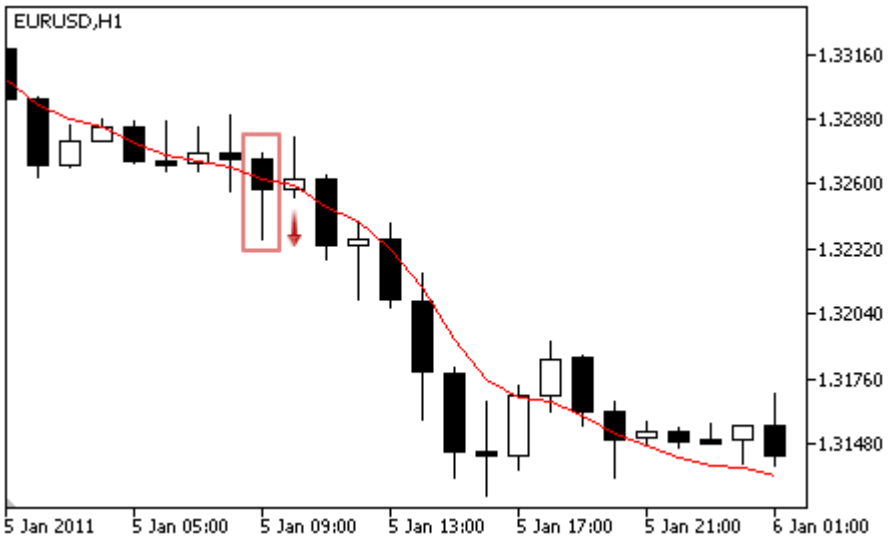
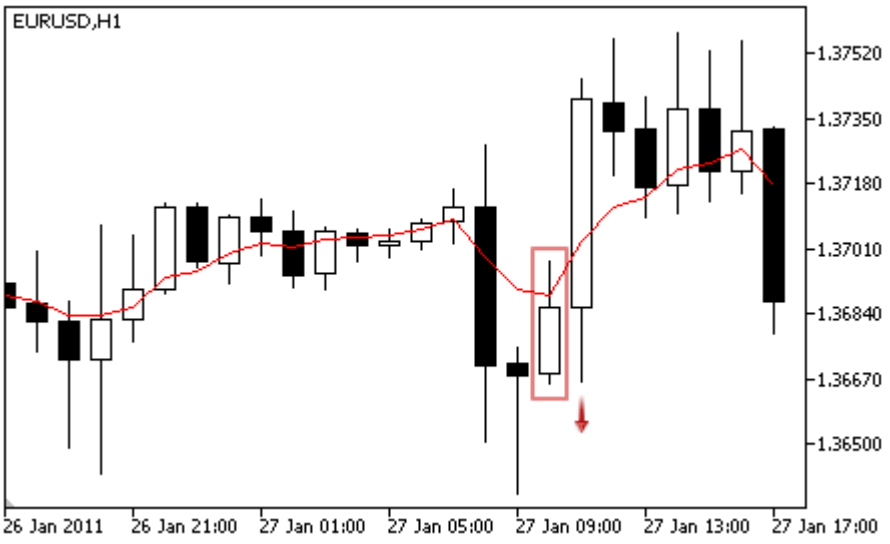
Questo modulo è basato sui modelli di mercato dell'indicatore Double Exponential Moving Average. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• Il prezzo ha incrociato l'indicatore verso il basso (il prezzo di Apertura della barra analizzata è sopra l'indicatore e il prezzo di Chiusura è sotto l'indicatore) e l'indicatore sale (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di apertura della barra analizzata è sotto l'indicatore ed il prezzo di chiusura è sopra l'indicatore) e l'indicatore sale (segnale forte). 

Tipo di segnale	Descrizione delle condizioni
	• L'ombra inferiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono al di sopra dell'indicatore, ed il prezzo Basso(Low) è sotto l'indicatore) e l'indicatore aumenta (segnale debole). 
Per il selling	• Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di Apertura(Open) della barra analizzata è sotto l'indicatore ed il prezzo di Chiusura(Close) è sopra l' indicatore) e l'indicatore scende (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso il basso (il prezzo di apertura(Open) della barra analizzata è sopra l'indicatore ed il prezzo di chiusura(Close) è sotto l'indicatore) e l'indicatore scende (segnale forte).

Tipo di segnale	Descrizione delle condizioni
	 • L'ombra superiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono sotto l'indicatore, ed il prezzo High è al di sopra dell' indicatore) e l'indicatore scende (segnale debole). 
Nessuna obiezione per il buying	Il prezzo è al di sopra dell'indicatore.
Nessuna obiezione alla vendita	Il prezzo è sotto l'indicatore.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

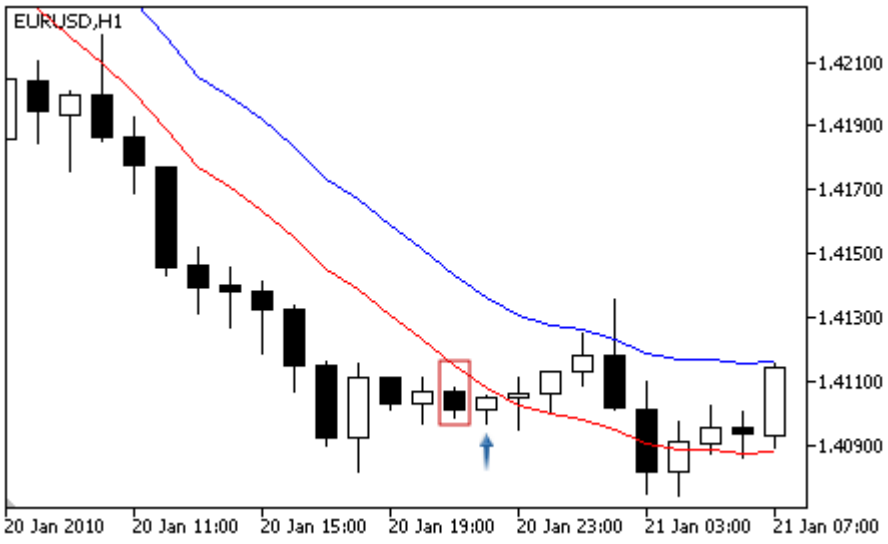
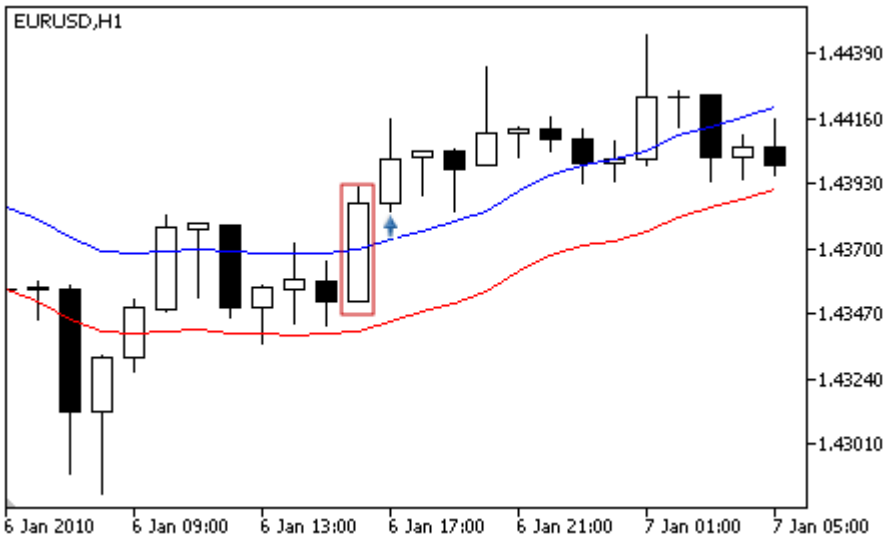
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodMA	Periodo di media dell'indicatore.
Shift	Slittamento dell'indicatore lungo l'asse temporale (in barre).
Metodo	Metodo di calcolo della media.
Applied	A price series usate per il calcolo dell'indicatore.

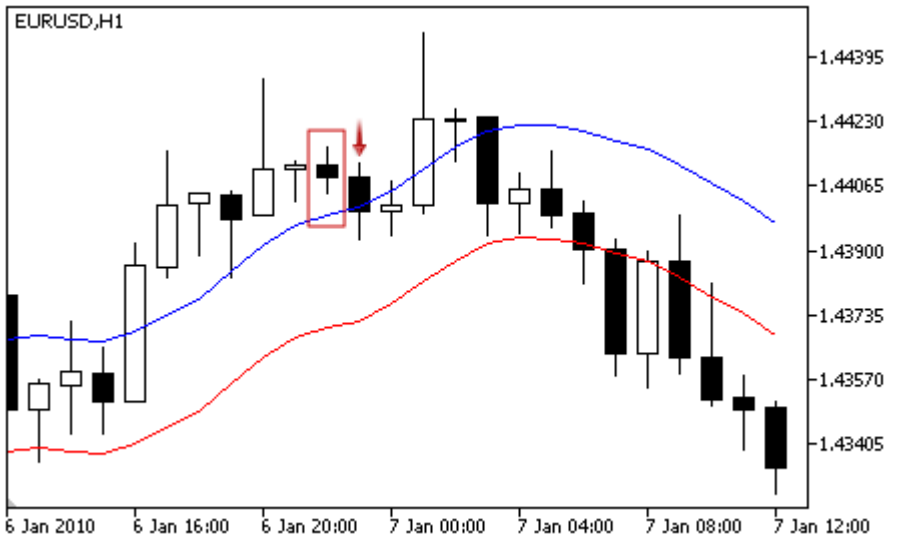
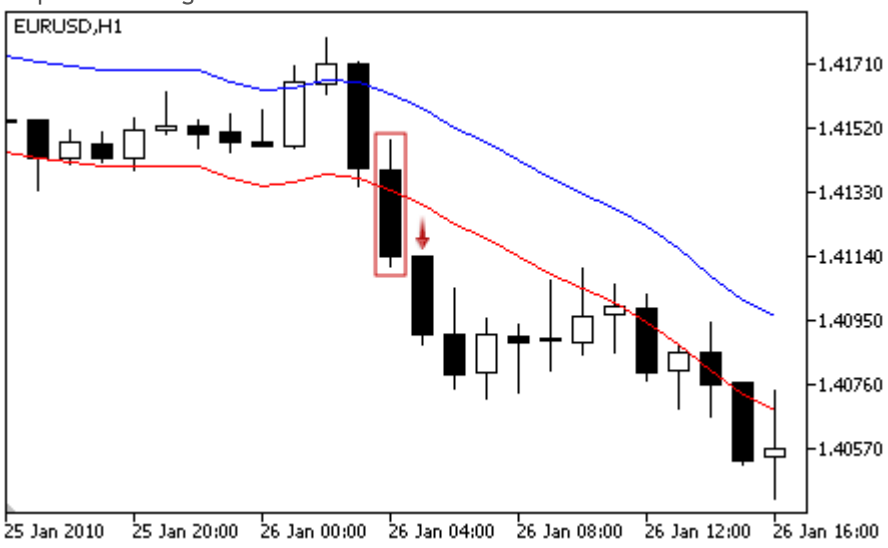
Segnali del indicatore Indicator Envelopes

Questo modulo di segnali si basa sui modelli di mercato dell'indicatore Envelopes. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• Il prezzo è vicino alla linea più bassa dell'indicatore alla barra analizzata.  • Il prezzo ha tagliato la linea superiore dell'indicatore alla barra analizzata. 
Per il selling	• Il prezzo è vicino alla linea superiore dell'indicatore alla barra analizzata.

Tipo di segnale	Descrizione delle condizioni
	 • Il prezzo ha tagliato la linea inferiore dell'indicatore alla barra analizzata. 
Nessuna obiezione per il buying	Nessun segnale.
Nessuna obiezione alla vendita	Nessun segnale.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

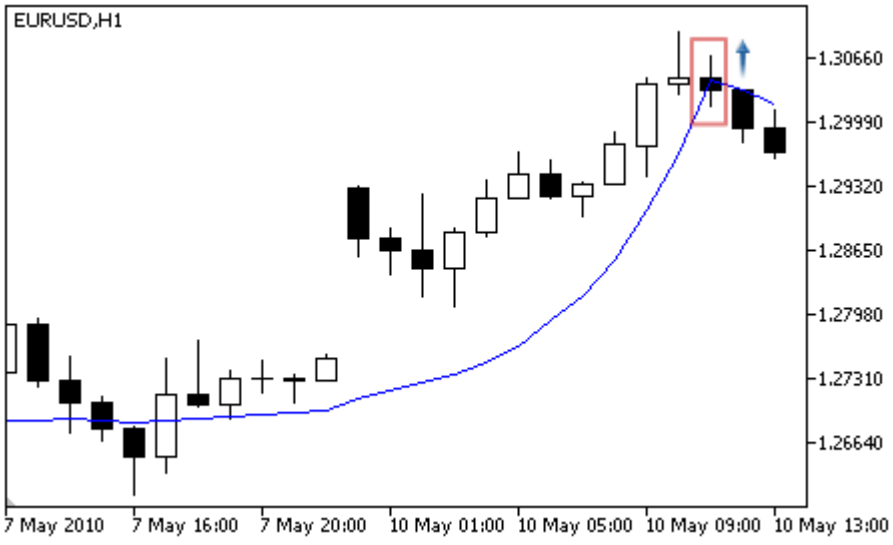
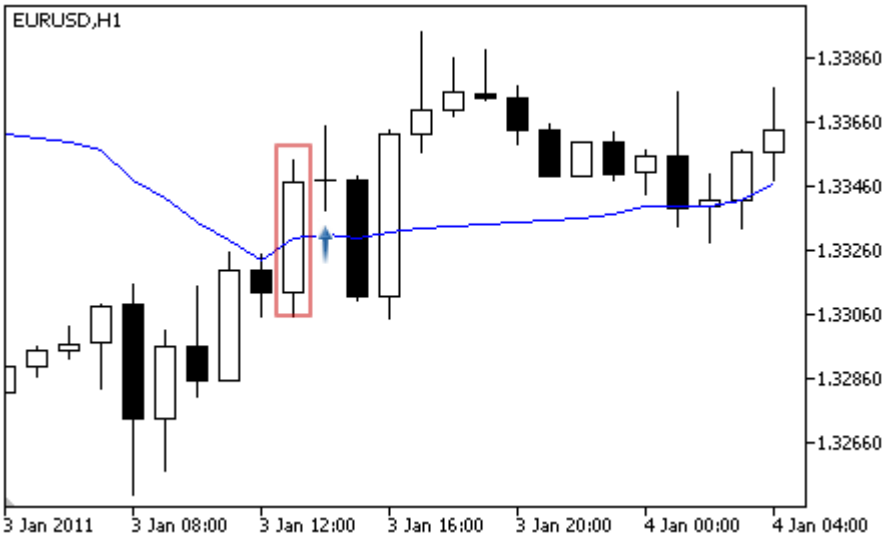
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodMA	Periodo di calcolo dell'indicatore.
Shift	Slittamento dell'indicatore lungo l'asse temporale (in barre).
Metodo	Metodo di calcolo della media.
Applied	A price series usate per il calcolo dell'indicatore.
Deviation	Deviazione dei bordi envelope dalla linea centrale (MA) in termini percentuali.

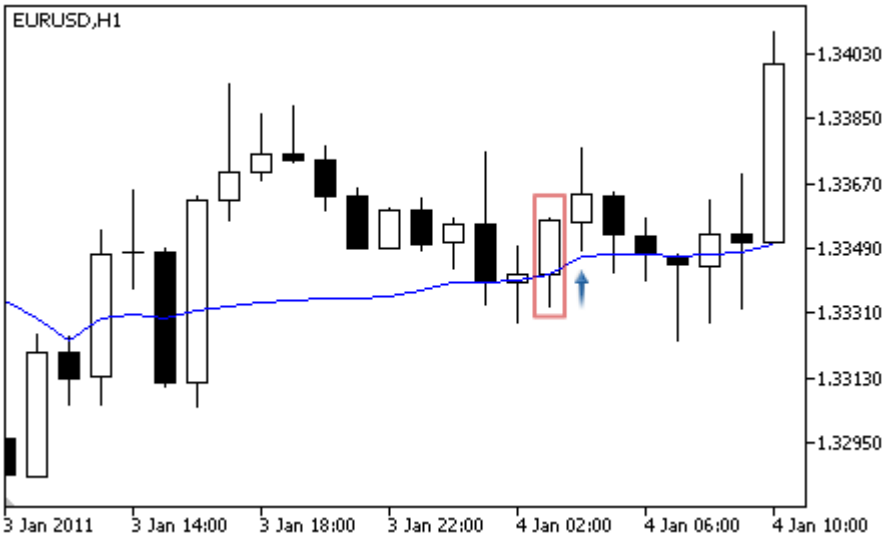
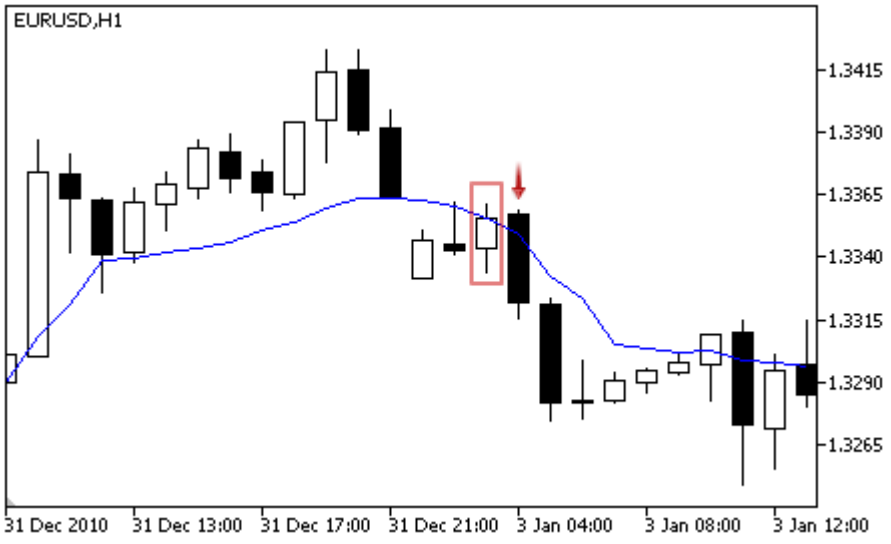
I segnali dell'indicatore Fractal Adaptive Moving Average

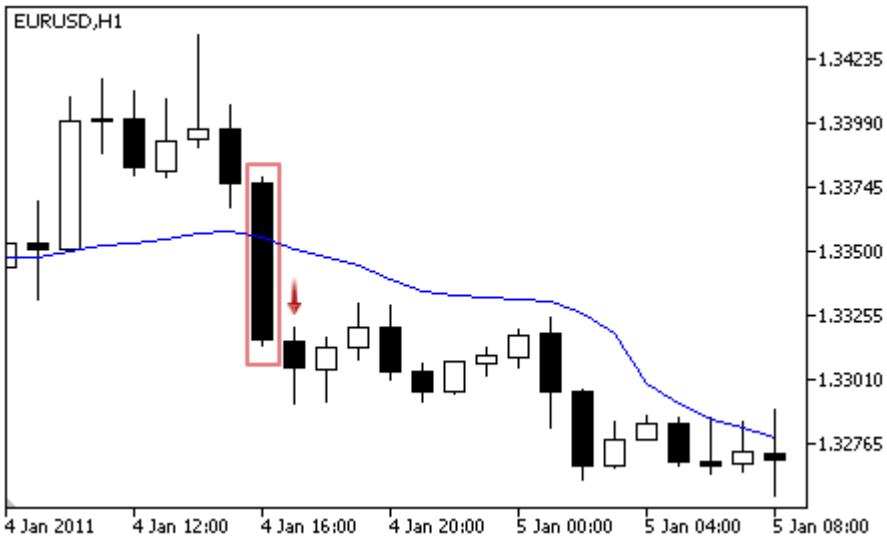
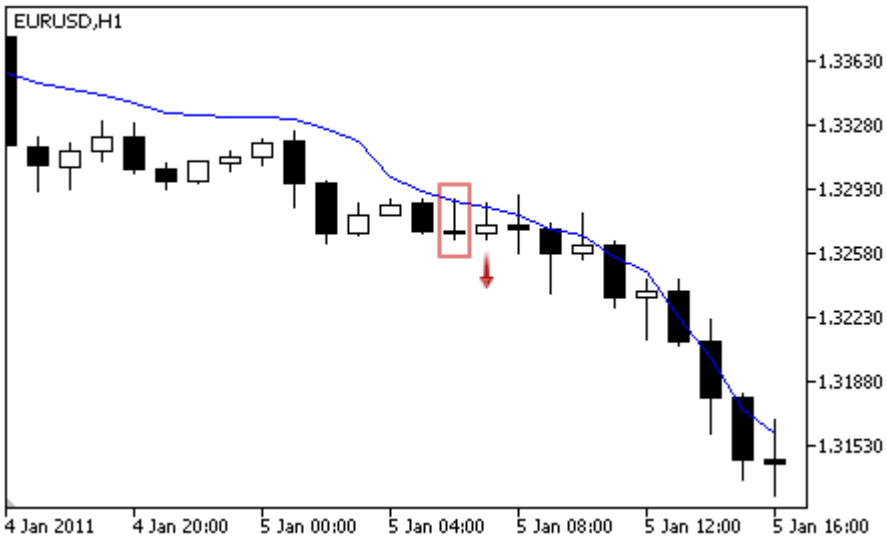
Questo modulo di segnali si basa sui modelli di mercato dell'indicatore Fractal Adaptive Moving Average. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• Il prezzo ha incrociato l'indicatore verso il basso (il prezzo di Apertura della barra analizzata è sopra l'indicatore e il prezzo di Chiusura è sotto l'indicatore) e l'indicatore sale (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di apertura della barra analizzata è sotto l'indicatore ed il prezzo di chiusura è sopra l'indicatore) e l'indicatore sale (segnale forte). 

Tipo di segnale	Descrizione delle condizioni
	• L'ombra inferiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono al di sopra dell'indicatore, ed il prezzo Basso(Low) è sotto l'indicatore) e l'indicatore aumenta (segnale debole). 
Per il selling	• Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di Apertura(Open) della barra analizzata è sotto l'indicatore ed il prezzo di Chiusura(Close) è sopra l' indicatore) e l'indicatore scende (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso il basso (il prezzo di apertura(Open) della barra analizzata è sopra l'indicatore ed il prezzo di chiusura(Close) è sotto l'indicatore) e l'indicatore scende (segnale forte).

Tipo di segnale	Descrizione delle condizioni
	 • L'ombra superiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono sotto l'indicatore, ed il prezzo High è al di sopra dell' indicatore) e l'indicatore scende (segnale debole). 
Nessuna obiezione per il buying	Il prezzo è sopra l'indicatore.
Nessuna obiezione alla vendita	Il prezzo è sotto l'indicatore.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodMA	Periodo di media dell'indicatore.
Shift	Slittamento dell'indicatore lungo l'asse temporale (in barre).
Metodo	Metodo di calcolo della media.
Applied	A price series usate per il calcolo dell'indicatore.

Segnali del Filtro Orario Intraday

Questo modulo si basa sul presupposto che l'efficienza di modelli di mercato cambia nel tempo. Utilizzando questo modulo, è possibile filtrare i segnali ricevuti dagli altri moduli per ore e giorni della settimana. Esso permette di aumentare la qualità dei segnali generati per via del taglio dei periodi di tempo sfavorevoli. Il meccanismo di presa di decisioni di trade basati sui segnali dei moduli è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	Nessun segnale.
Per il selling	Nessun segnale.
Nessuna obiezione per il buying	La data e l'ora correnti soddisfano i parametri specificati.
Nessuna obiezione alla vendita	La data e l'ora correnti soddisfano i parametri specificati.

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
GoodHourOfDay	Numero di unica ora del giorno (da 0 a 23) quando verranno abilitati i segnali di trade. Se il valore è -1, i segnali saranno abilitati per l'intera giornata.
BadHoursOfDay	Il campo bit. Ogni bit di questo campo corrisponde ad un'ora del giorno (0 bit - ora 0, ..., 23 bit - 23-esima ora). Se il valore di un bit è uguale a 0, i segnali di trade saranno attivati nella corrispondente ora. Se il valore di un bit è uguale a 1, i segnali di trade saranno disabilitati durante le corrispondenti ore. Un numero specificato è rappresentato come un numero binario e viene utilizzato come maschera di bit. Ore disabilitate hanno una priorità più alta rispetto a quelle abilitate.
GoodDayOfWeek	Numero dell'unico giorno della settimana (da 0 a 6, dove 0 è la Domenica), quando saranno abilitati segnali di trade. Se il valore è -1, i segnali saranno abilitati in qualsiasi giorno.
BadDaysOfWeek	Il campo bit. Ogni bit di questo campo corrisponde ad un giorno della settimana (0 bit - Domenica, ..., 6 bit - Sabato). Se il valore di un bit è uguale a 0, i segnali di trade saranno attivati nel corrispondente giorno. Se il valore di un bit è uguale a 1, i segnali di trade saranno disabilitati

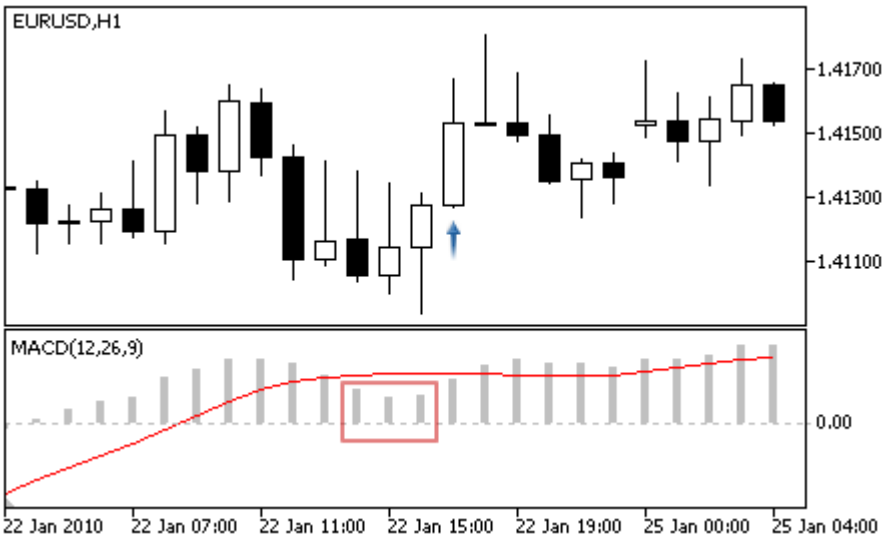
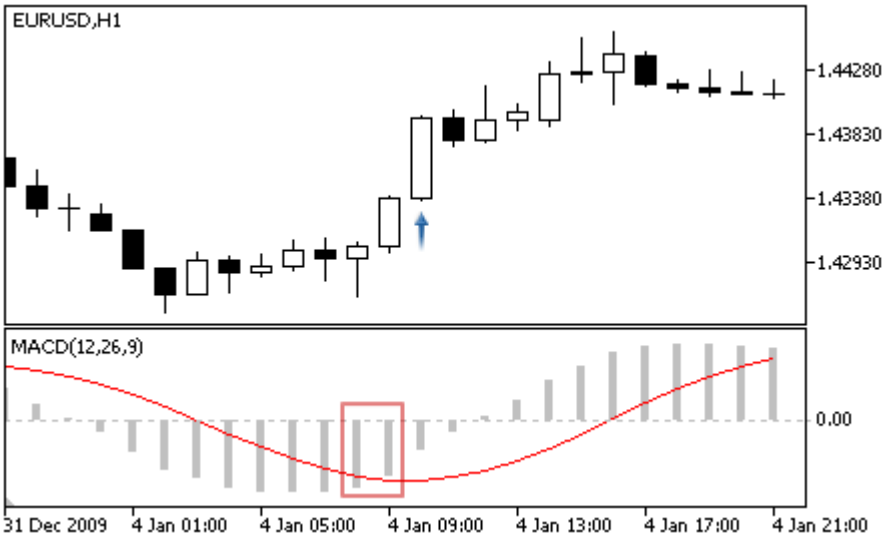
Parametro	Descrizione
	durante il corrispondente giorno. Un numero specificato è rappresentato come un numero binario e viene utilizzato come maschera di bit. Giorni disabilitati hanno una priorità più alta rispetto a quelli abilitati.

I segnali del MACD Oscillator

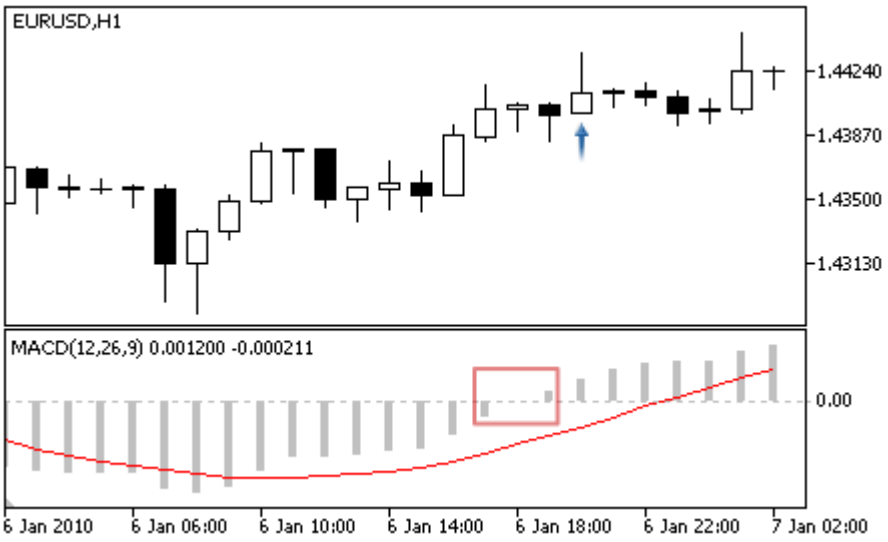
Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore MACD. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

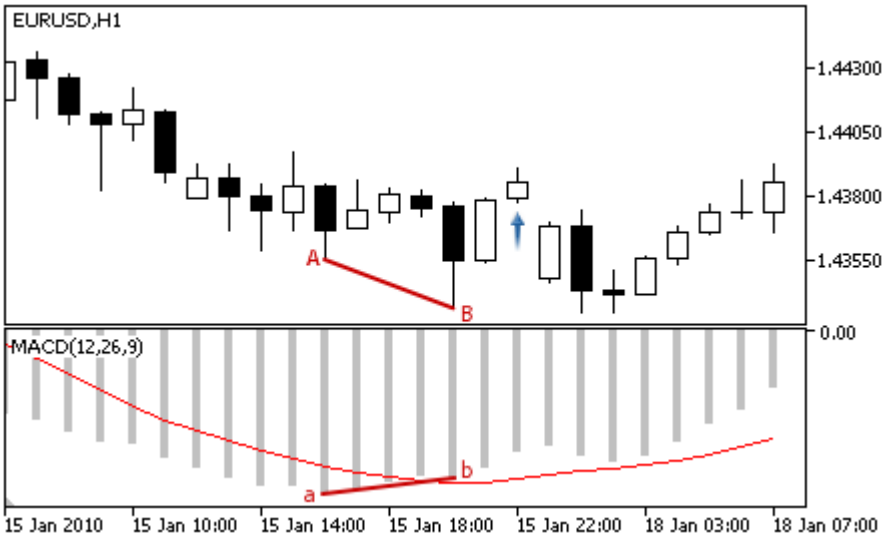
Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Reverse** — 'oscillatore si rivolta verso l'alto (l'oscillatore sale alla barra analizzata e cade alla precedente).  • **Crossover della linea principale(main) e segnale(signal** — la linea principale è sopra la linea segnale alla barra analizzata e sotto la linea segnale alla barra precedente.  • **Crossing del livello zero** — la linea principale è sopra il livello zero alla barra analizzata e sotto il livello zero alla barra precedente.

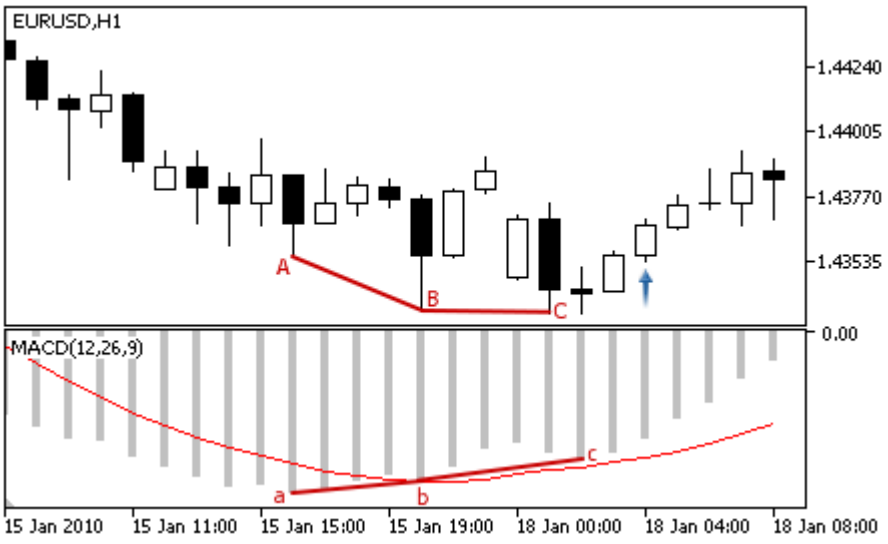
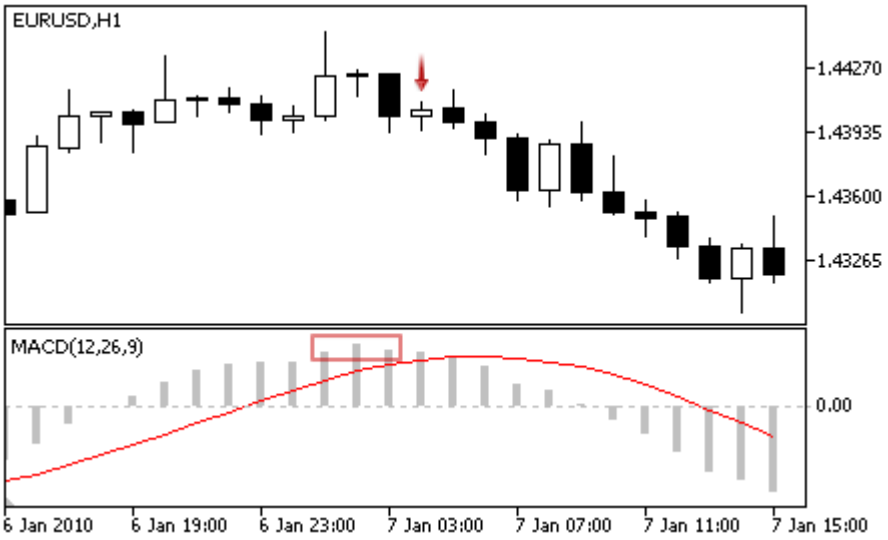
Tipo di segnale	Descrizione delle condizioni



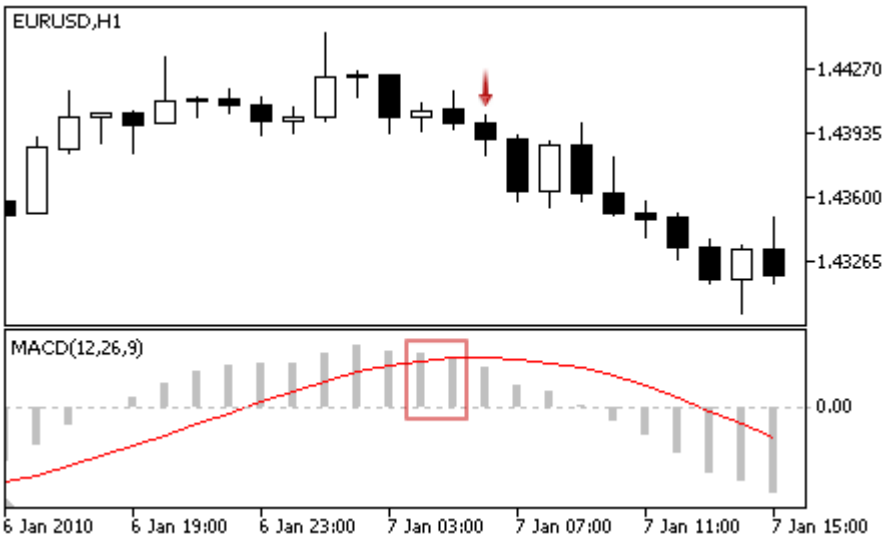
- **Divergenza** — il primo fondo analizzato dell'oscillatore è superiore a quello precedente, ed il corrispondente fondo(in basso) prezzo è inferiore al precedente.



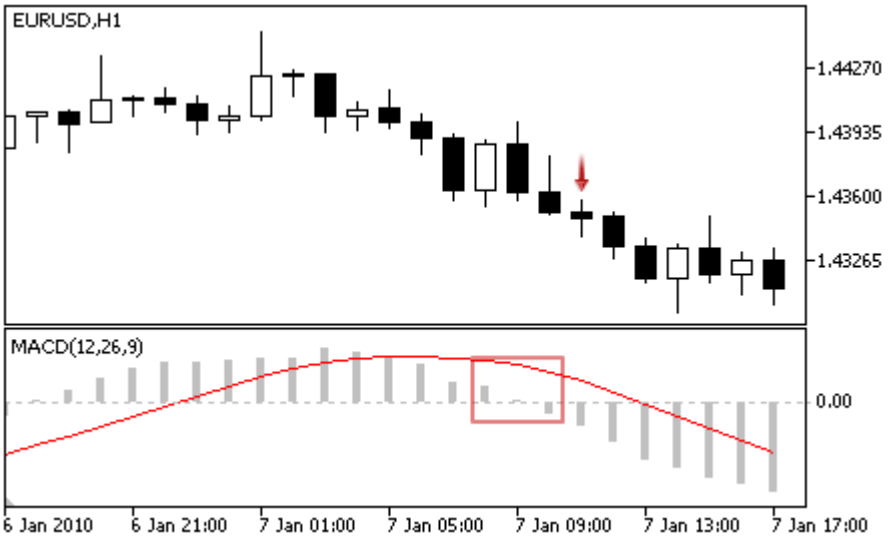
- **Doppia divergenza(double divergence)** - l' oscillatore forma tre conseguenti fondi, ciascuno di loro è superiore al precedente; ed il prezzo ha formato tre fondi corrispondenti, e ciascuno di essi è inferiore a quello precedente.

Tipo di segnale	Descrizione delle condizioni
	
Per il selling	• **Reverse** — l'oscillatore si rivolta verso il basso (l'oscillatore cade alla barra analizzata e sorge alla precedente).  • **Crossover della linea principale(main) e segnale(signal** — la linea principale è sotto la linea segnale alla barra analizzata e sopra la linea segnale alla barra precedente.

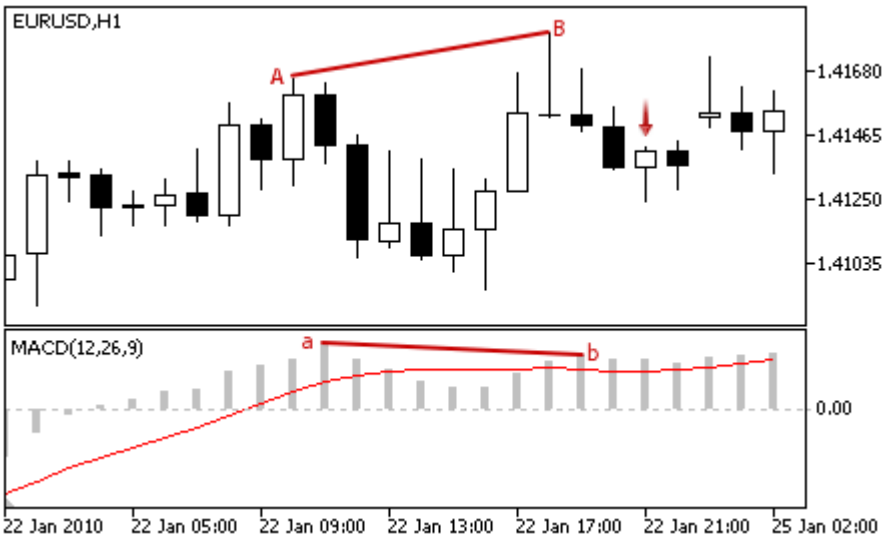
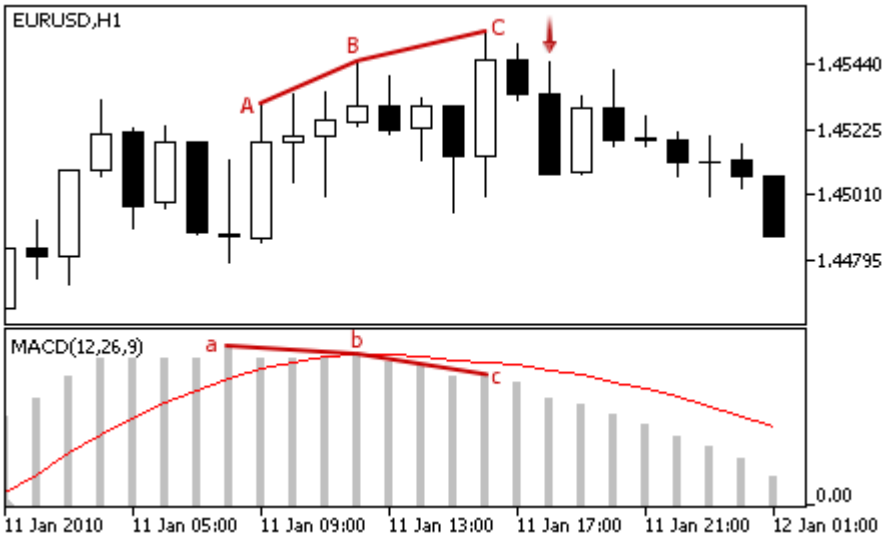
Tipo di segnale	Descrizione delle condizioni



- **Incrociando il livello zero** — la linea principale è inferiore al livello zero alla barra analizzata e sopra il livello zero alla precedente.



- **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente ed il prezzo picco corrispondente è superiore al picco precedente.

Tipo di segnale	Descrizione delle condizioni
	 • **Doppia divergenza** — l'oscillatore forma tre conseguenti picchi, ciascuno di loro è inferiore a quello precedente; il prezzo forma tre picchi corrispondenti, ciascuno dei quali è superiore a quello precedente. 
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

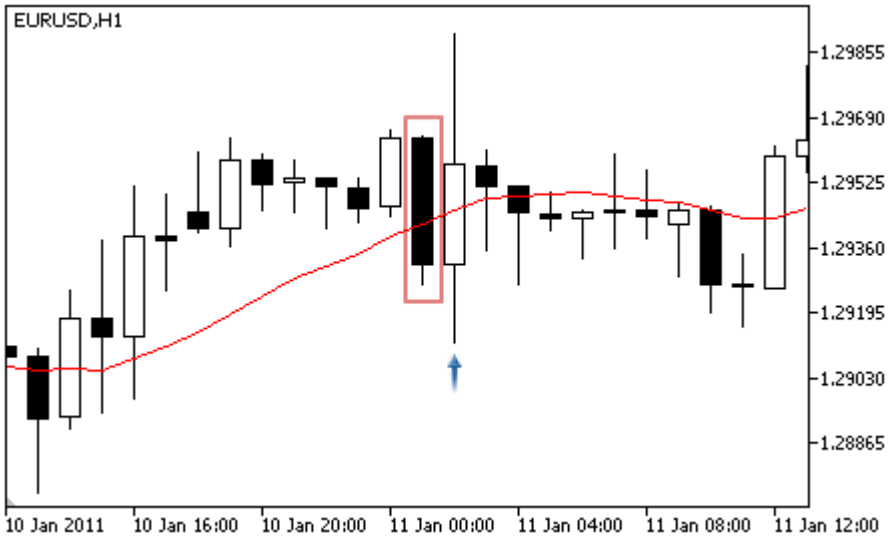
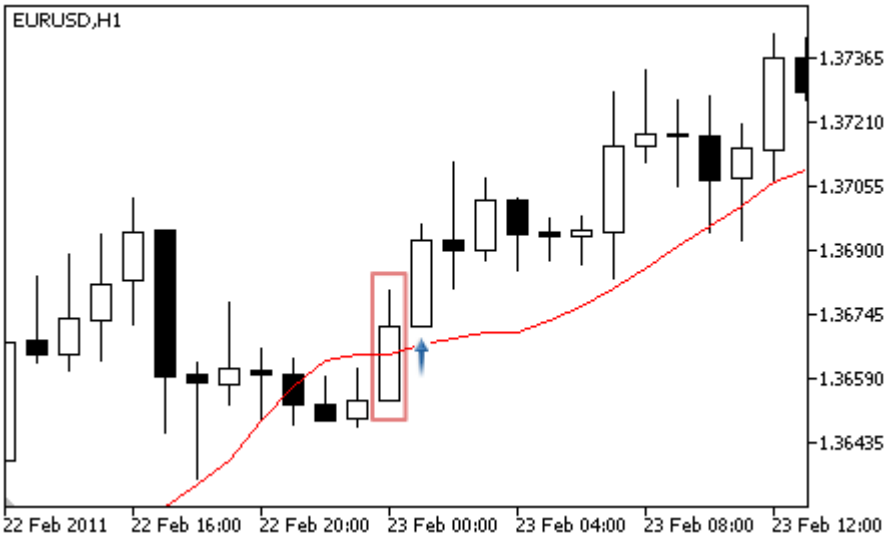
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodFast	Periodo di calcolo della EMA veloce.
PeriodSlow	Periodo di calcolo della EMA lenta.
PeriodSignal	Periodo di smoothing.
Applied	Una serie di prezzi utilizzata per il calcolo dell'oscillatore.

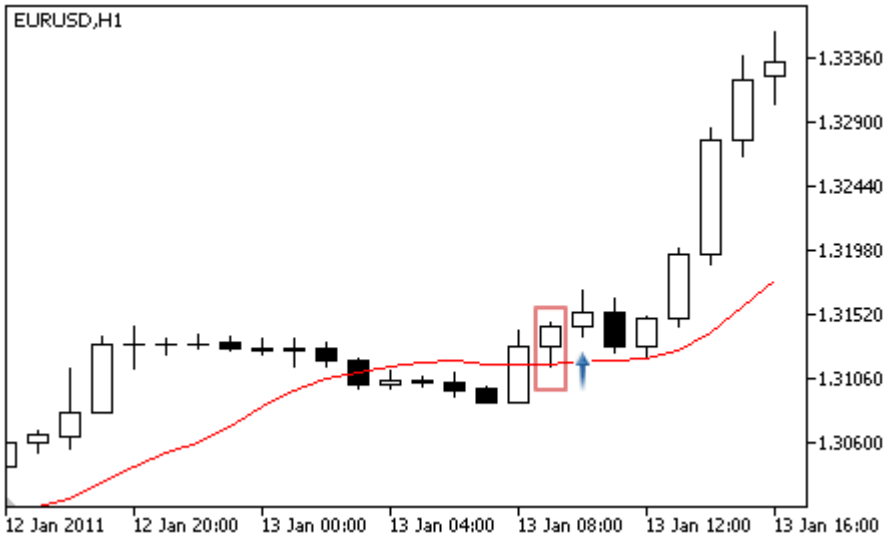
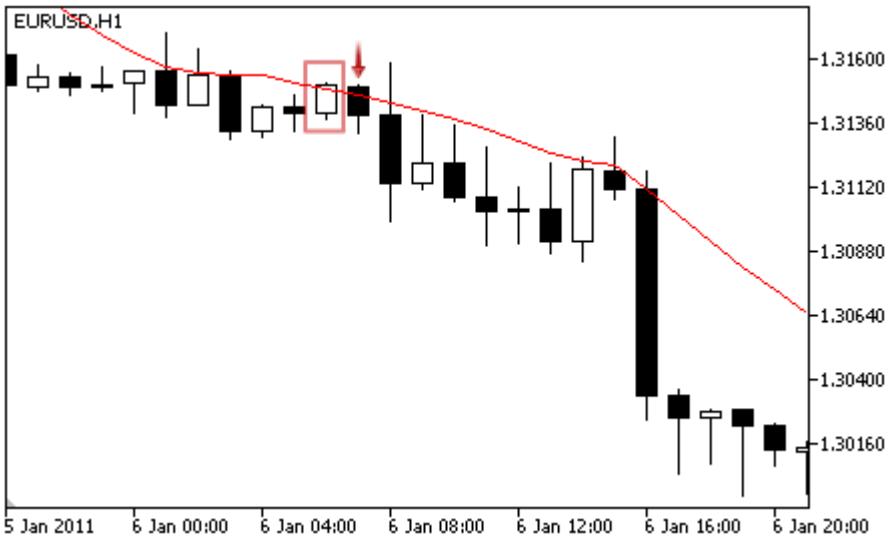
I segnali dell'indicatore Moving Average

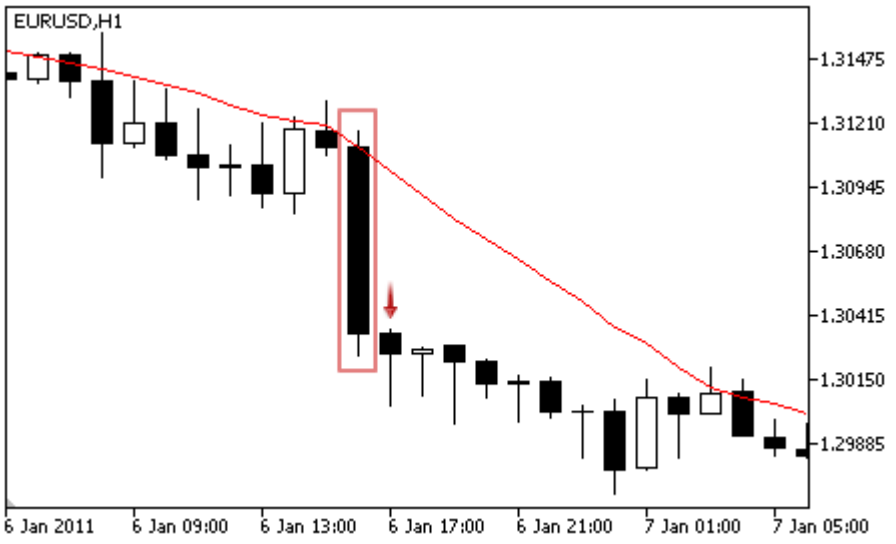
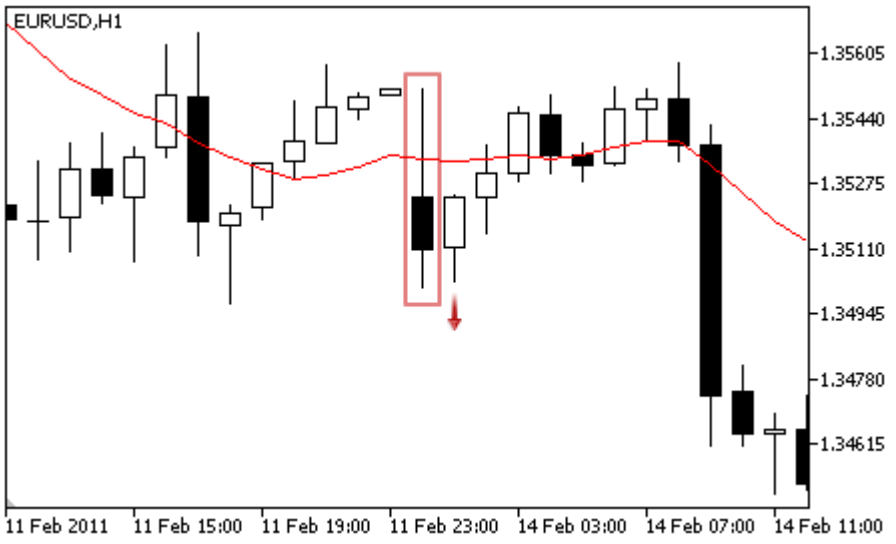
Questo modulo di segnali si basa sui modelli di mercato dell'indicatore Moving Average. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• Il prezzo ha incrociato l'indicatore verso il basso (il prezzo di Apertura della barra analizzata è sopra l'indicatore e il prezzo di Chiusura è sotto l'indicatore) e l'indicatore sale (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di apertura della barra analizzata è sotto l'indicatore ed il prezzo di chiusura è sopra l'indicatore) e l'indicatore sale (segnale forte). 

Tipo di segnale	Descrizione delle condizioni
	• L'ombra inferiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono al di sopra dell'indicatore, ed il prezzo Basso(Low) è sotto l'indicatore) e l'indicatore aumenta (segnale debole). 
Per il selling	• Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di Apertura(Open) della barra analizzata è sotto l'indicatore ed il prezzo di Chiusura(Close) è sopra l' indicatore) e l'indicatore scende (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso il basso (il prezzo di apertura(Open) della barra analizzata è sopra l'indicatore ed il prezzo di chiusura(Close) è sotto l'indicatore) e l'indicatore scende (segnale forte).

Tipo di segnale	Descrizione delle condizioni
	 • L'ombra superiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono sotto l'indicatore, ed il prezzo High è al di sopra dell' indicatore) e l'indicatore scende (segnale debole). 
Nessuna obiezione per il buying	Il prezzo è sopra l'indicatore.
Nessuna obiezione alla vendita	Il prezzo è sotto l'indicatore.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

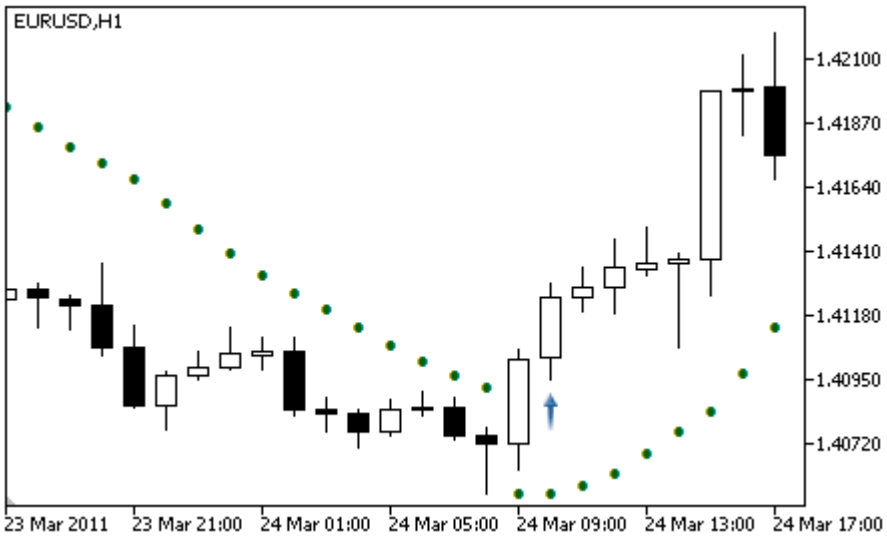
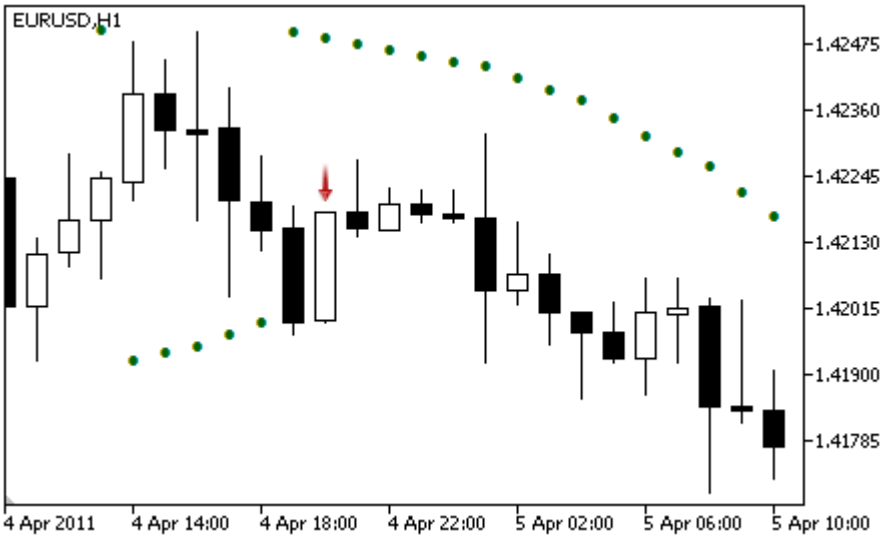
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodMA	Periodo di media dell'indicatore.
Shift	Slittamento dell'indicatore lungo l'asse temporale (in barre).
Metodo	Metodo di calcolo della media.
Applied	A price series usate per il calcolo dell'indicatore.

Segnali dell'indicatore Parabolic SAR

Questo modulo di segnali si basa sui modelli di mercato dell'indicatore Parabolic SAR. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	**Reverse** — l'indicatore è inferiore al prezzo alla barra analizzata e sopra il prezzo a quella precedente. 
Per il selling	**Reverse** — l'indicatore è sopra al prezzo alla barra analizzata e sotto al prezzo a quella precedente. 
Nessuna obiezione per il buying	Il prezzo è sopra l'indicatore.

Tipo di segnale	Descrizione delle condizioni
Nessuna obiezione alla vendita	Il prezzo è sotto l'indicatore.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

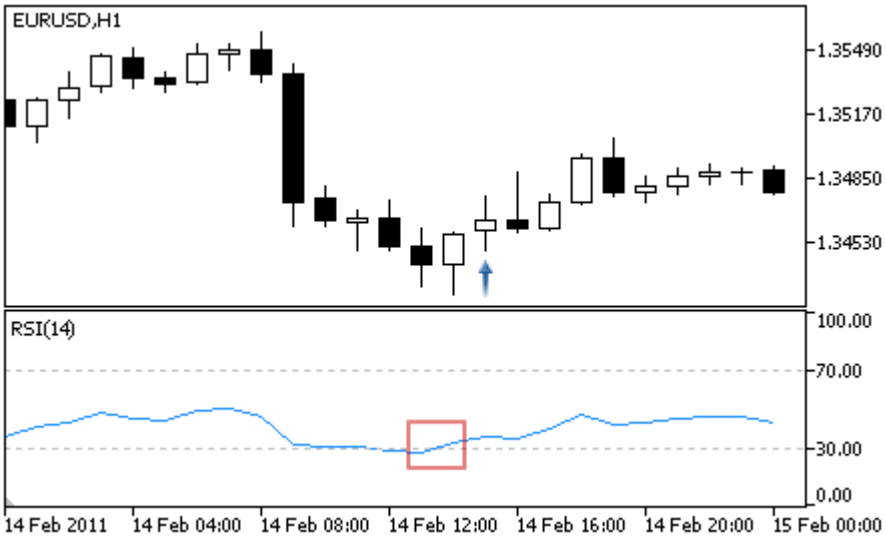
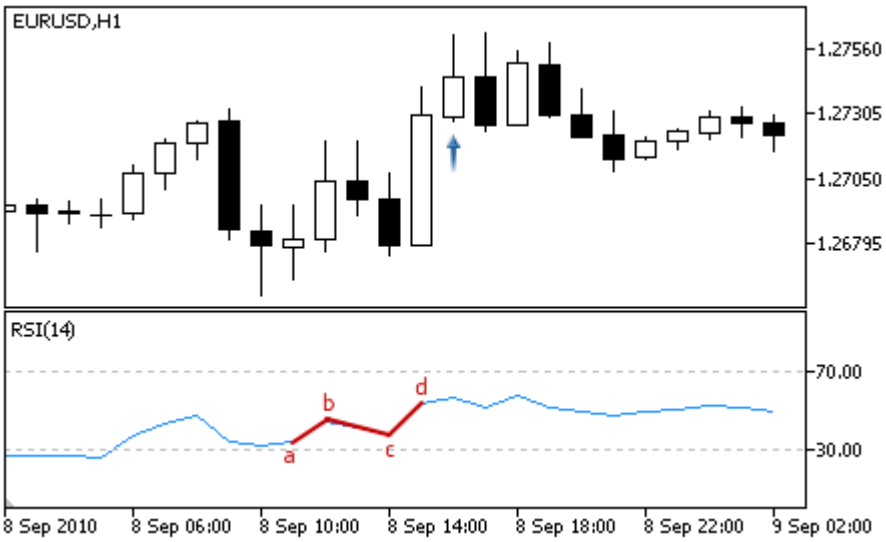
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
Step	Lo step di incremento dei prezzi.
Maximum	Il tasso massimo della velocità di convergenza dell'indicatore con il prezzo.

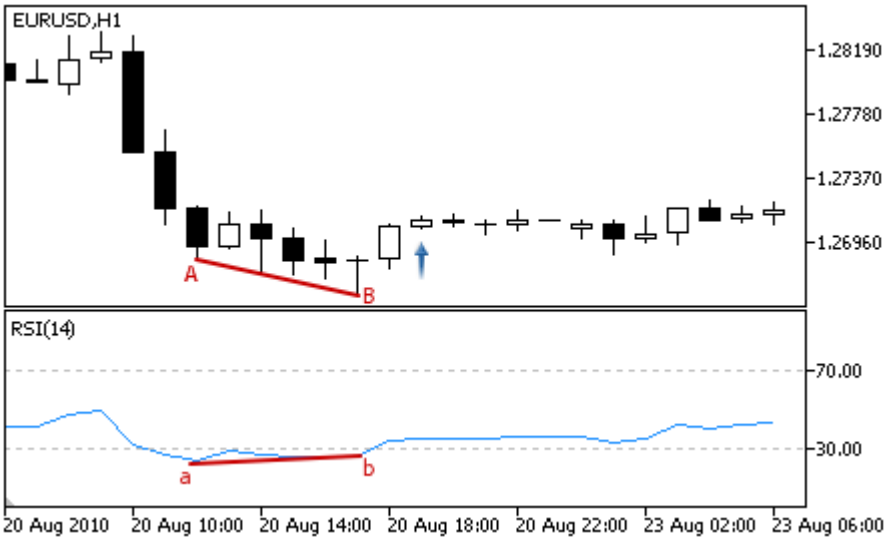
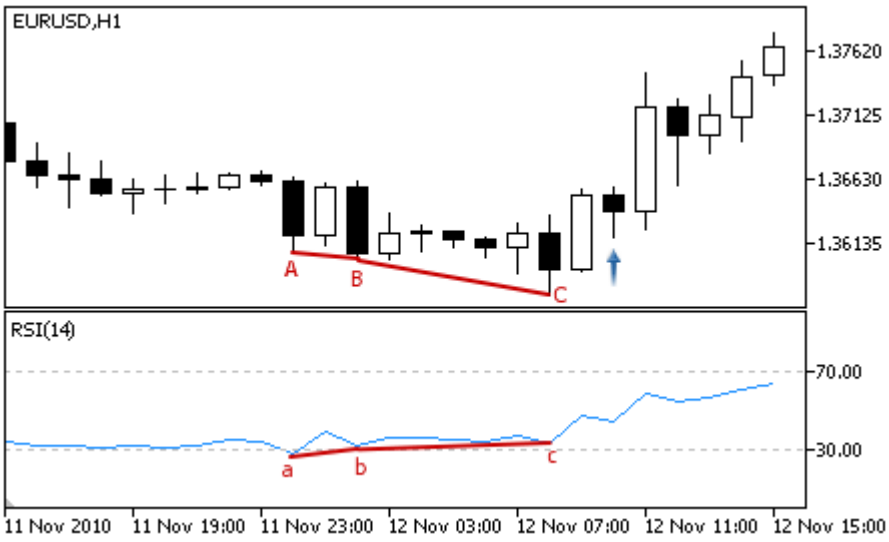
I segnali dell'oscillatore Relative Strength Index

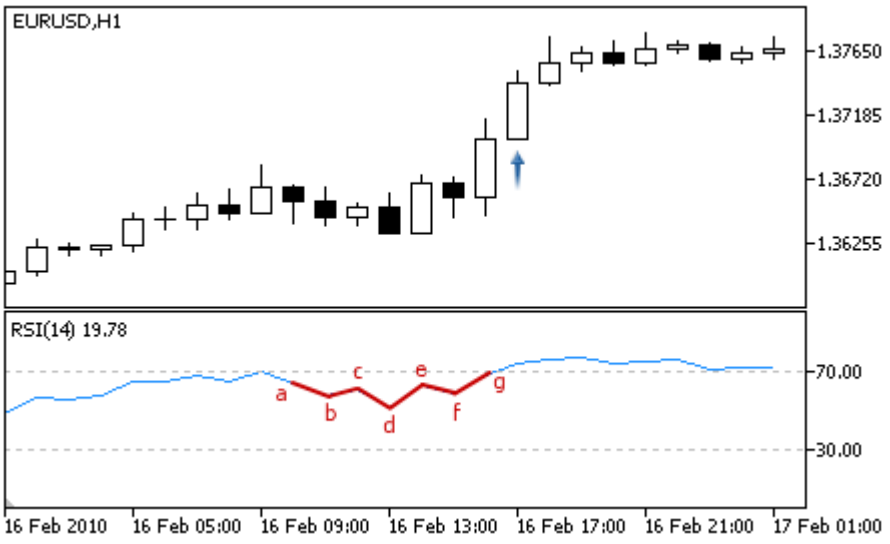
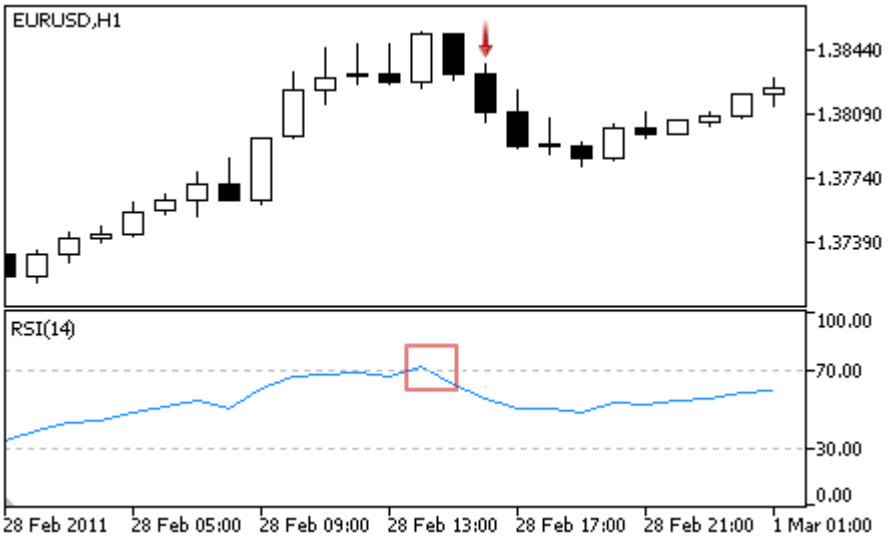
Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore Relative Strength Index. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

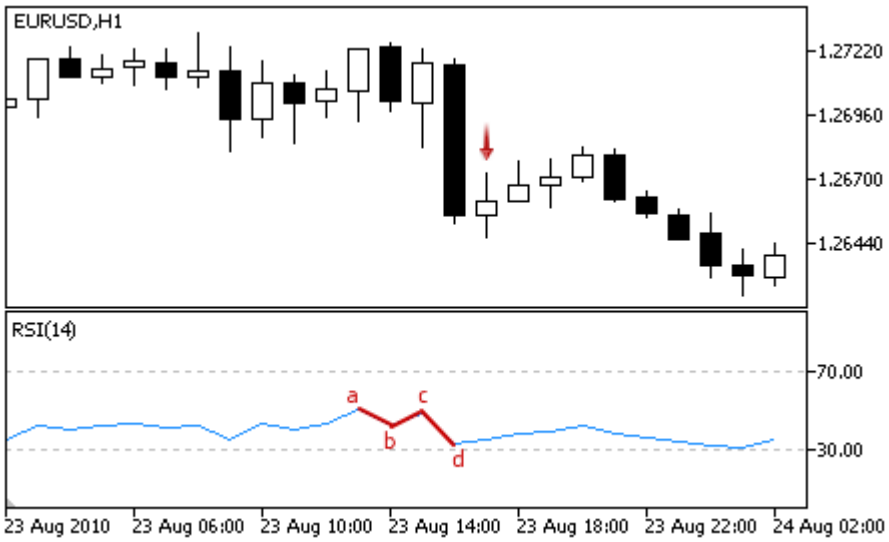
Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Reverse dietro il livello di oversold** — l'oscillatore si rivolta verso l'alto e il suo valore alla barra analizzata è dietro il livello di oversold (valore di default è 30).  • **Swing fallito** — l'oscillatore sale più alto del precedente picco alla barra analizzata. 

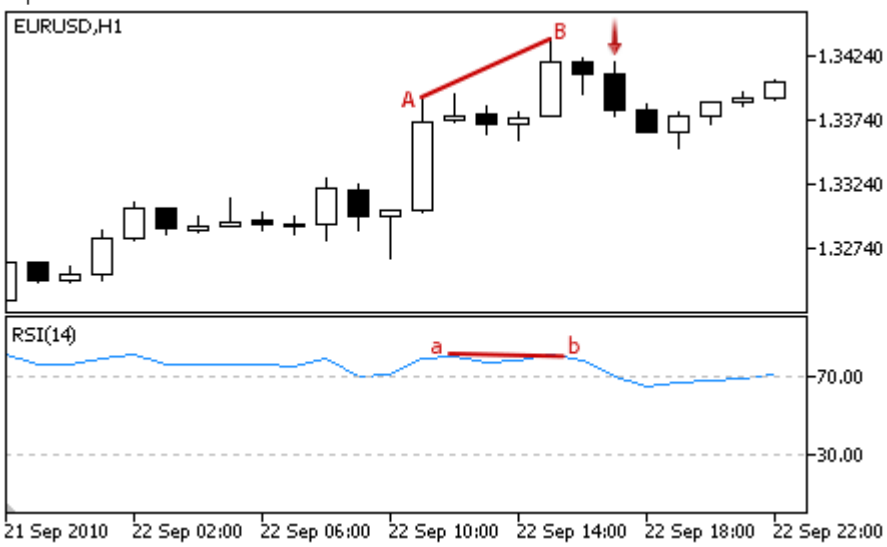
Tipo di segnale	Descrizione delle condizioni
	• **Divergenza** — il primo fondo analizzato dell'oscillatore è è inferiore a quello precedente, ed il corrispondente fondo prezzo bottom è inferiore al precedente.  • **Doppia divergenza(double divergence)** - l' oscillatore forma tre conseguenti fondi, ciascuno di loro è superiore al precedente; ed il prezzo ha formato tre fondi corrispondenti, e ciascuno di essi è inferiore a quello precedente.  • **Testa/Spalle** — l'oscillatore ha formato tre conseguenti fondi, e quello in mezzo è inferiore rispetto agli altri.

Tipo di segnale	Descrizione delle condizioni
	
Per il selling	• **Reverse dietro il livello di ipercomprato** — l'oscillatore si rivolta verso il basso e il suo valore alla barra analizzata è dietro il livello di ipercomprato (valore di default è 70).  • **Swing fallito** — l'oscillatore scende al di sotto del fondo precedente alla barra analizzata.

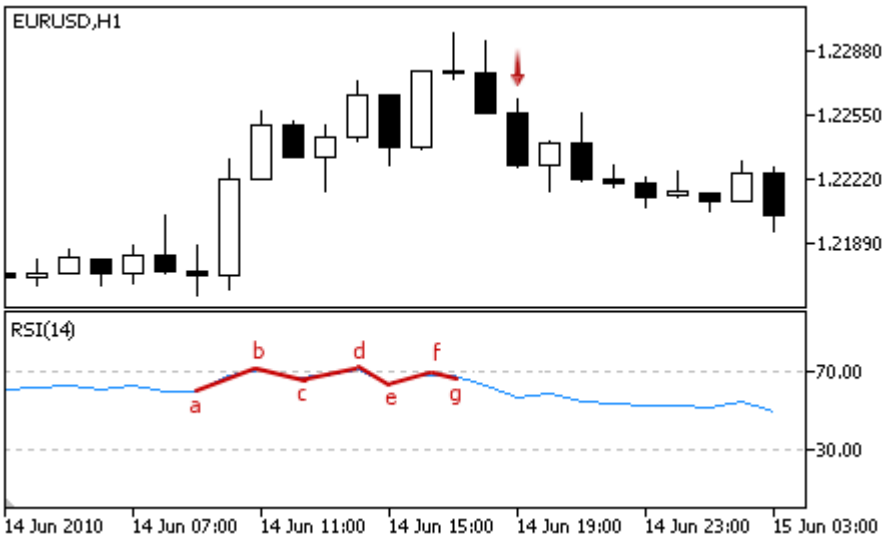
Tipo di segnale	Descrizione delle condizioni



- **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente ed il prezzo picco corrispondente è superiore al picco precedente.



- **Doppia divergenza** — l'oscillatore forma tre conseguenti picchi, ciascuno di loro è inferiore a quello precedente; il prezzo forma tre picchi corrispondenti, ciascuno dei quali è superiore a quello precedente.

Tipo di segnale	Descrizione delle condizioni
	 • **Testa/Spalle** — l'oscillatore ha formato tre conseguenti picchi, e quell in mezzo è superiore rispetto agli altri. 
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

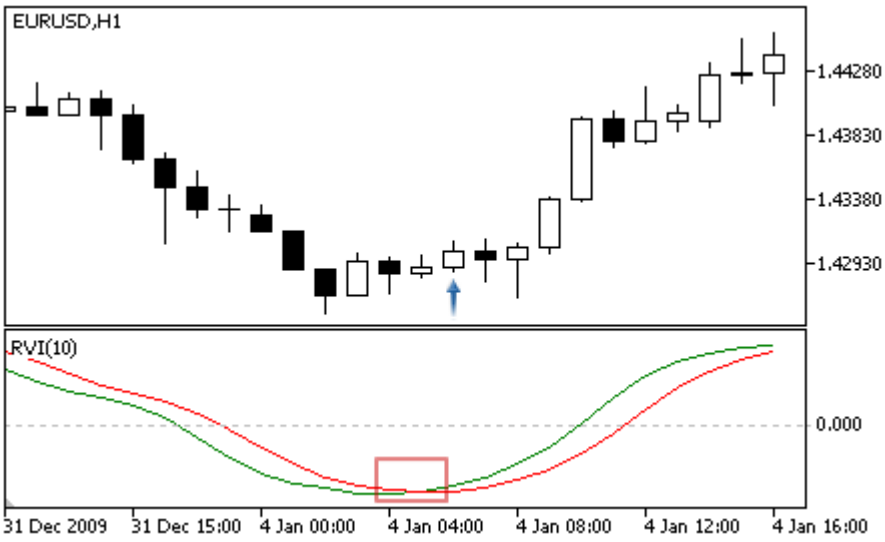
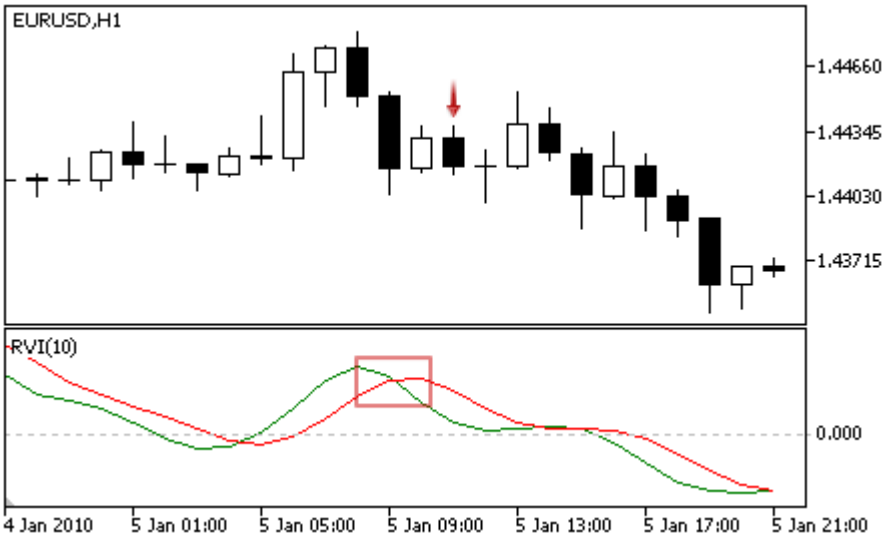
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodRSI	Periodo di calcolo dell'oscillatore.
Applied	Una serie di prezzi utilizzata per il calcolo dell'oscillatore.

I segnali dell'oscillatore Relative Vigor Index

Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore Relative Vigor Index. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	**Incrocio della linea principale e della segnale** — la linea principale è sopra la linea del segnale alla barra analizzata e sotto la linea di segnale alla precedente. 
Per il selling	**Incrocio della linea principale e segnale** — la linea principale è sotto la linea di segnale alla barra analizzata e sopra la linea di segnale precedente. 

Tipo di segnale	Descrizione delle condizioni
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

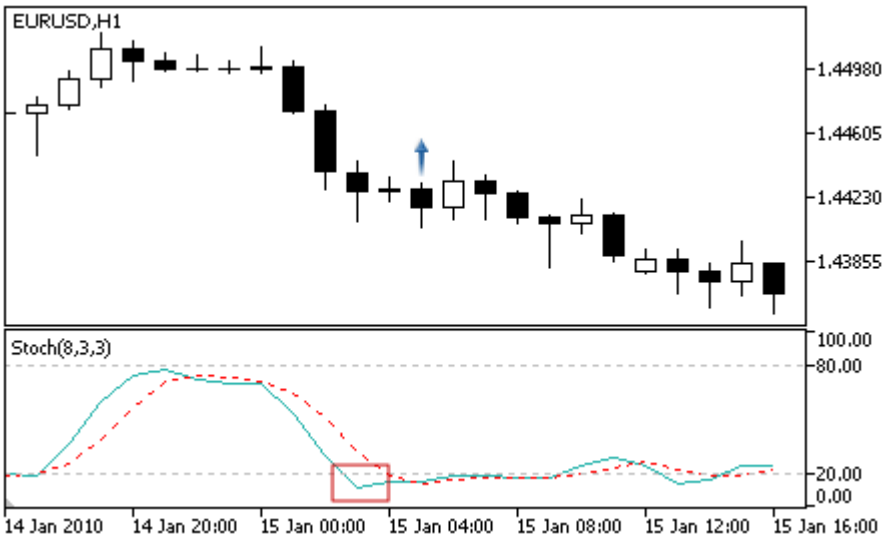
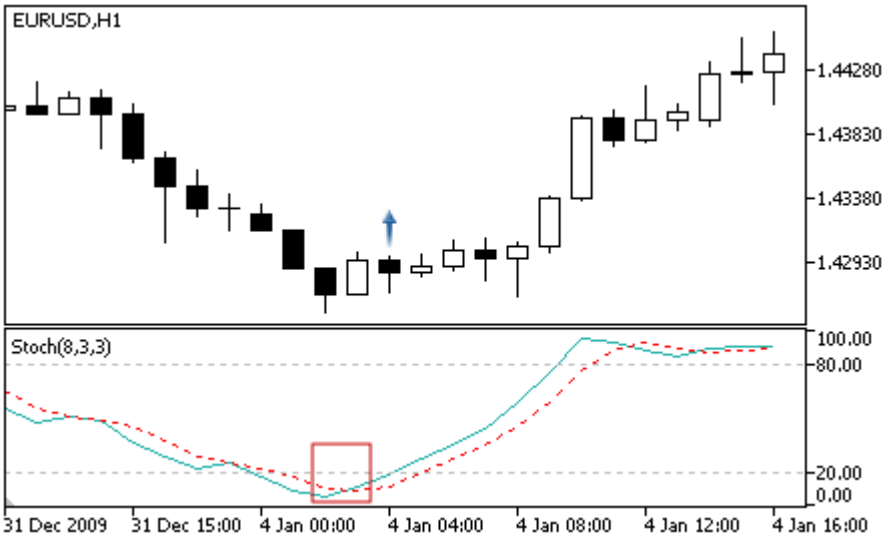
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodRVI	Periodo di calcolo dell'oscillatore.

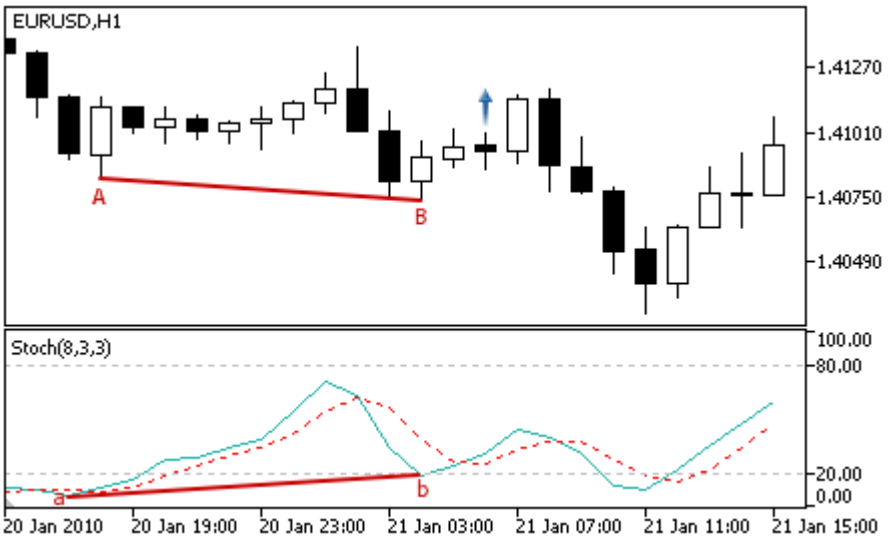
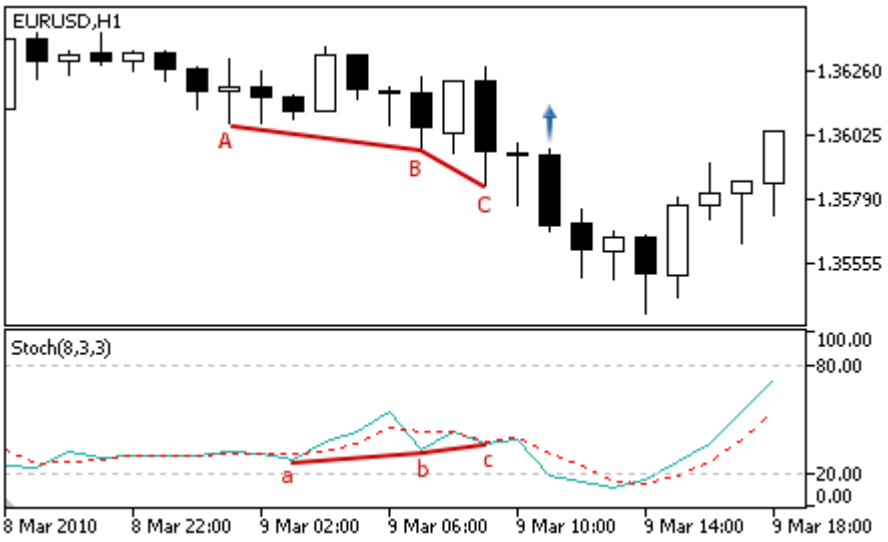
I segnali dell'Oscillatore Stochastic

Questo modulo è dei segnali sulla base dei modelli di mercato dell'oscillatore Stochastic. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

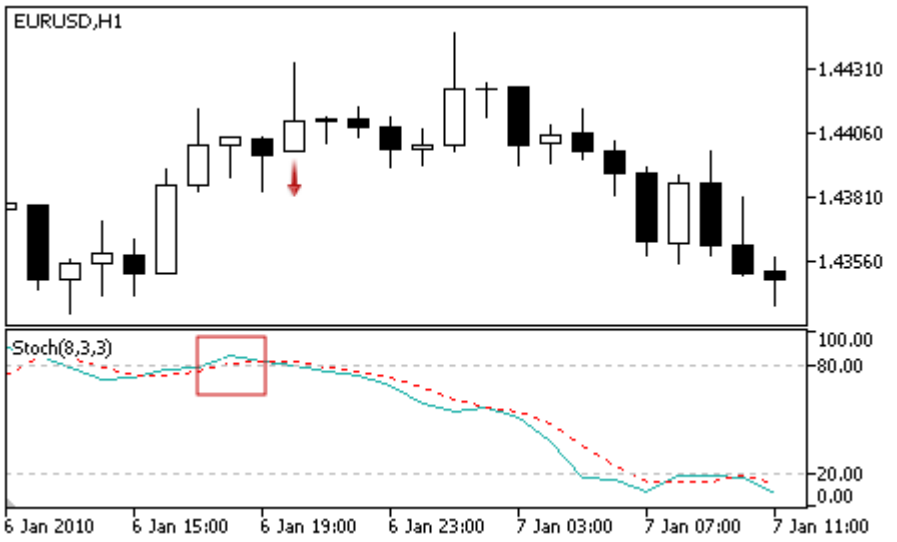
Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

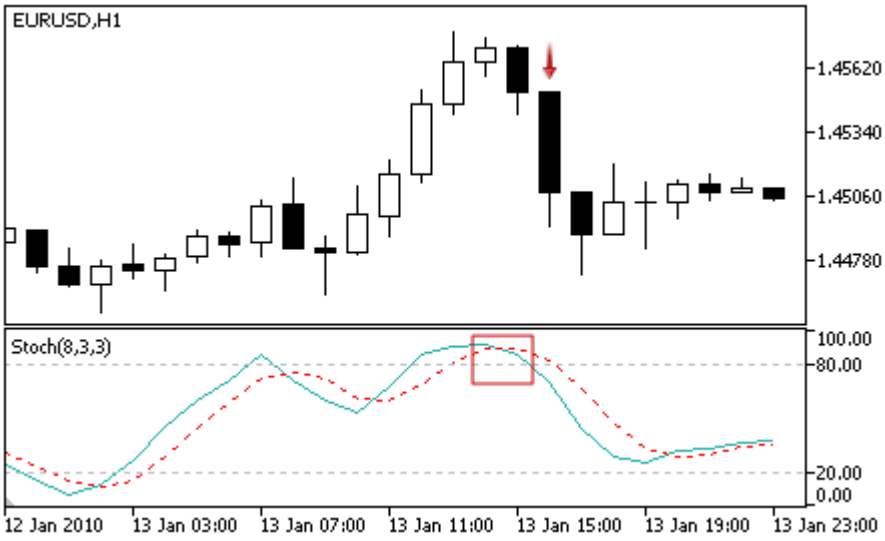
Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Reverse** — 'oscillatore si rivolta verso l'alto (l'oscillatore sale alla barra analizzata e cade alla precedente).  • **Incrocio della linea principale e della segnale** — la linea principale è sopra la linea del segnale alla barra analizzata e sotto la linea di segnale alla precedente. 

Tipo di segnale	Descrizione delle condizioni
	• **Divergenza** — il primo fondo analizzato dell'oscillatore è superiore a quello precedente, ed il corrispondente fondo(in basso) prezzo è inferiore al precedente.  • **Doppia divergenza(double divergence)** - l' oscillatore forma tre conseguenti fondi, ciascuno di loro è superiore al precedente; ed il prezzo ha formato tre fondi corrispondenti, e ciascuno di essi è inferiore a quello precedente. 
Per il selling	• **Reverse** — l'oscillatore si rivolta verso il basso (l'oscillatore cade alla barra analizzata e sorge alla precedente).

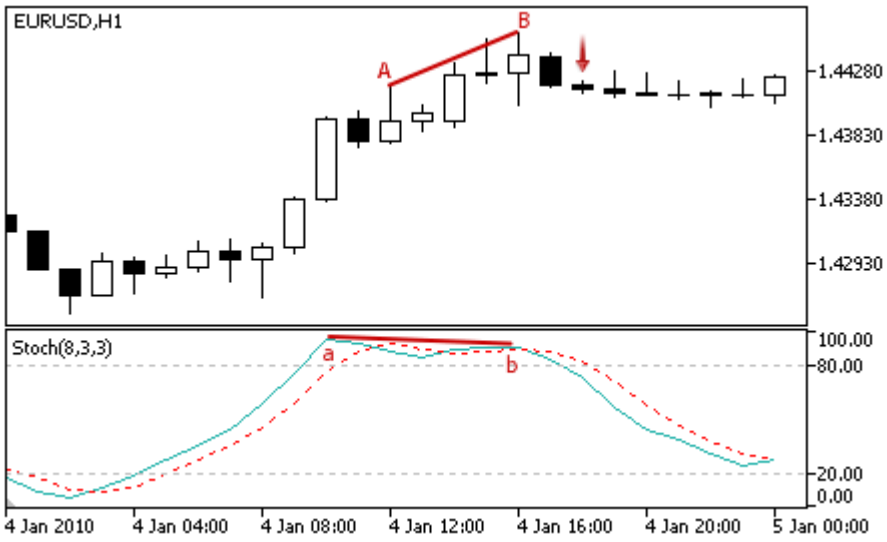
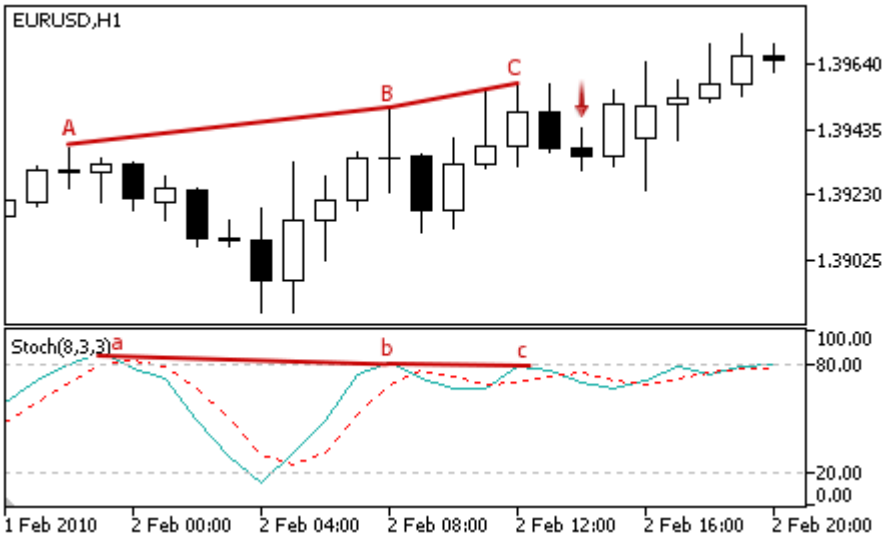
Tipo di segnale	Descrizione delle condizioni



- **Incrocio della linea principale e segnale** — la linea principale è sotto la linea di segnale alla barra analizzata e sopra la linea di segnale precedente.



- **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente ed il prezzo picco corrispondente è superiore al picco precedente.

Tipo di segnale	Descrizione delle condizioni
	 • **Doppia divergenza** — l'oscillatore forma tre conseguenti picchi, ciascuno di loro è inferiore a quello precedente; il prezzo forma tre picchi corrispondenti, ciascuno dei quali è superiore a quello precedente. 
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

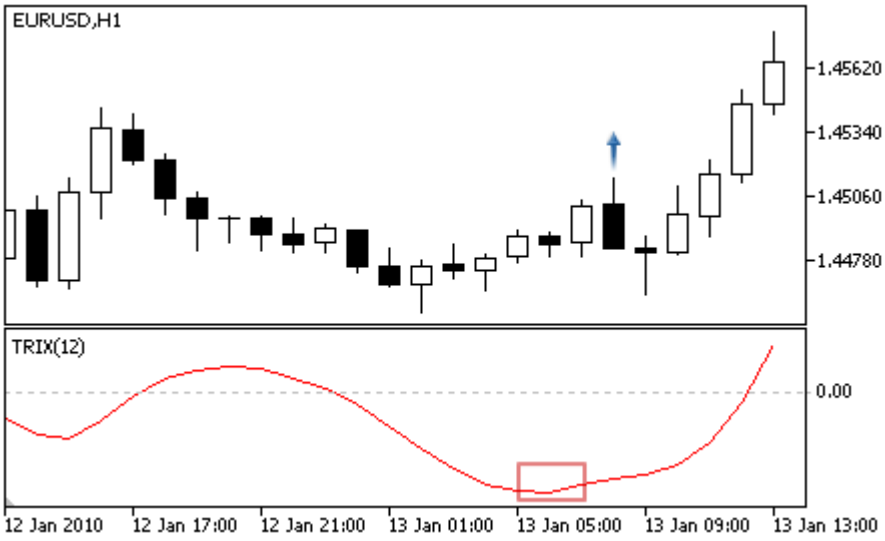
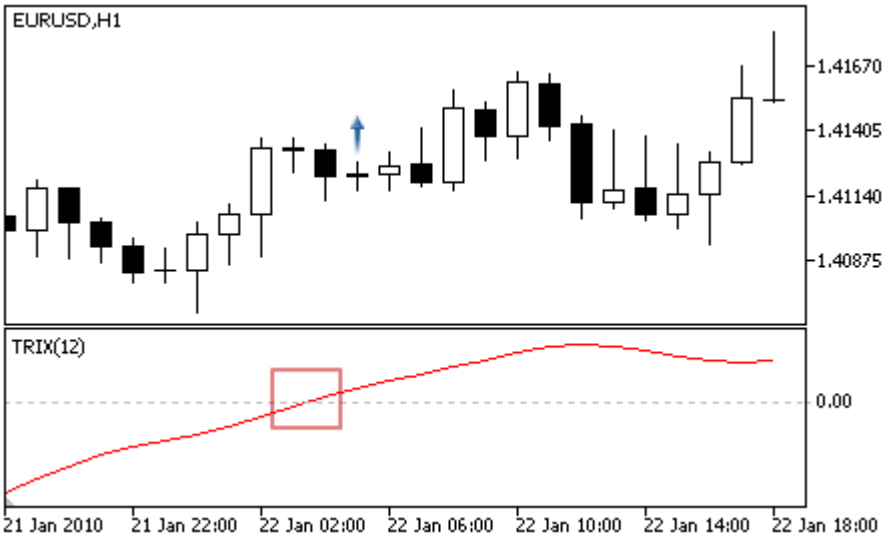
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodK	Periodo di calcolo della linea principale dell'oscillatore.
PeriodD	Periodo di calcolo della linea principale dell'oscillatore.
PeriodSlow	Periodo di rallentamento.
Applied	Una serie di prezzi utilizzata per il calcolo dell'oscillatore.

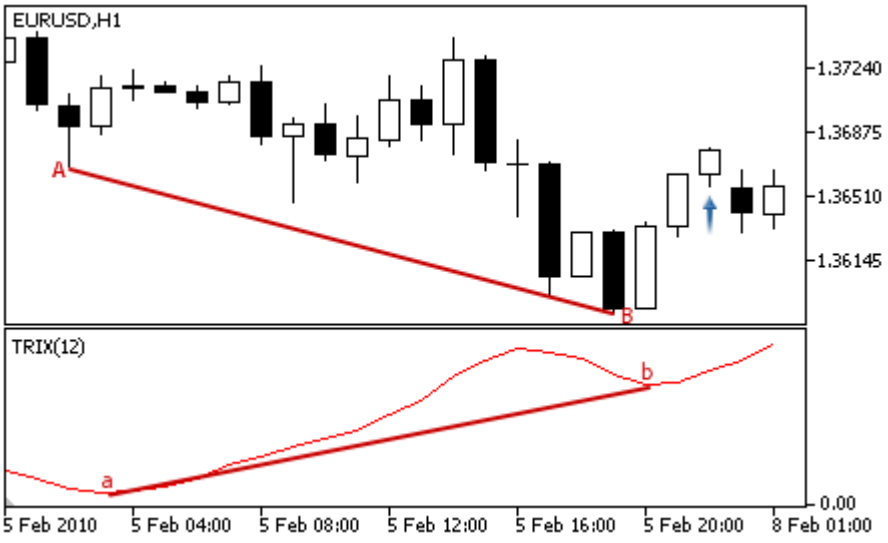
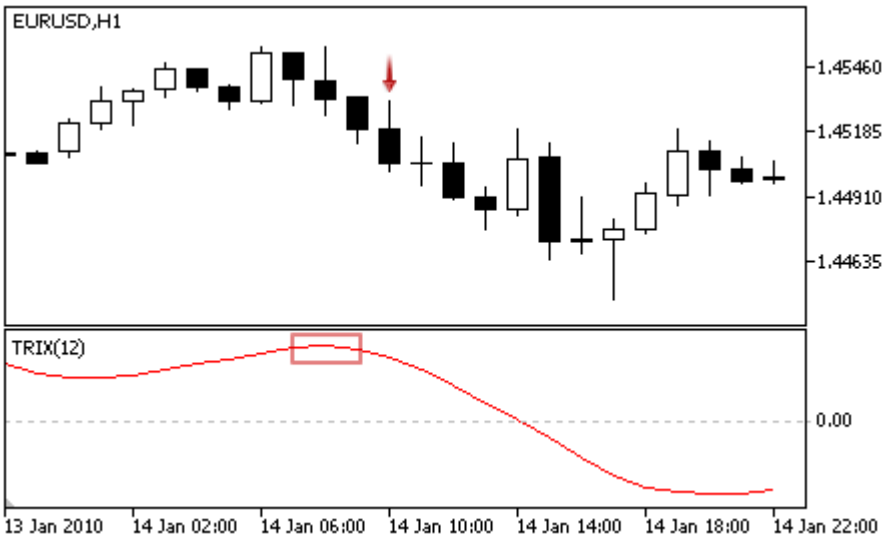
I segnali dell'oscillatore Triple Exponential Average

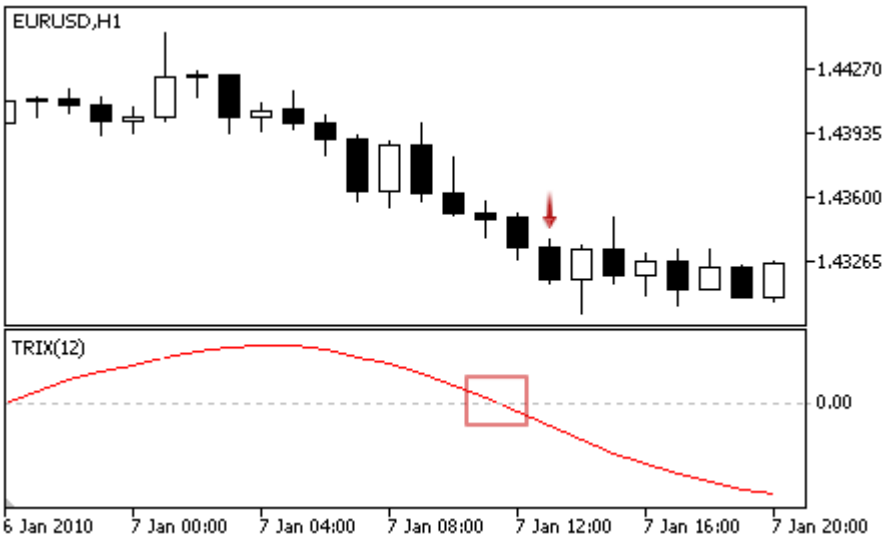
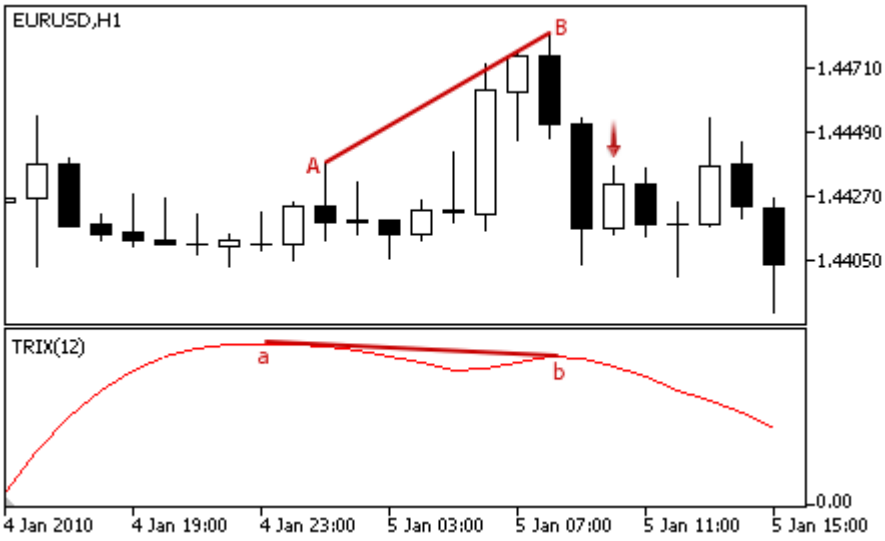
Questo modulo è dei segnali sulla base dei modelli di mercato dell'oscillatore Triple Exponential Average. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Reverse** — 'oscillatore si rivolta verso l'alto (l'oscillatore sale alla barra analizzata e cade alla precedente).  • **Crossing del livello zero** — la linea principale è sopra il livello zero alla barra analizzata e sotto il livello zero alla barra precedente. 

Tipo di segnale	Descrizione delle condizioni
	• **Divergenza** — il primo fondo analizzato dell'oscillatore è superiore a quello precedente, ed il corrispondente fondo(in basso) prezzo è inferiore al precedente. 
Per il selling	• **Reverse** — l'oscillatore si rivolta verso il basso (l'oscillatore cade alla barra analizzata e sorge alla precedente).  • **Incrociando il livello zero** — la linea principale è inferiore al livello zero alla barra analizzata e sopra il livello zero alla precedente.

Tipo di segnale	Descrizione delle condizioni
	 • **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente ed il prezzo picco corrispondente è superiore al picco precedente. 
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

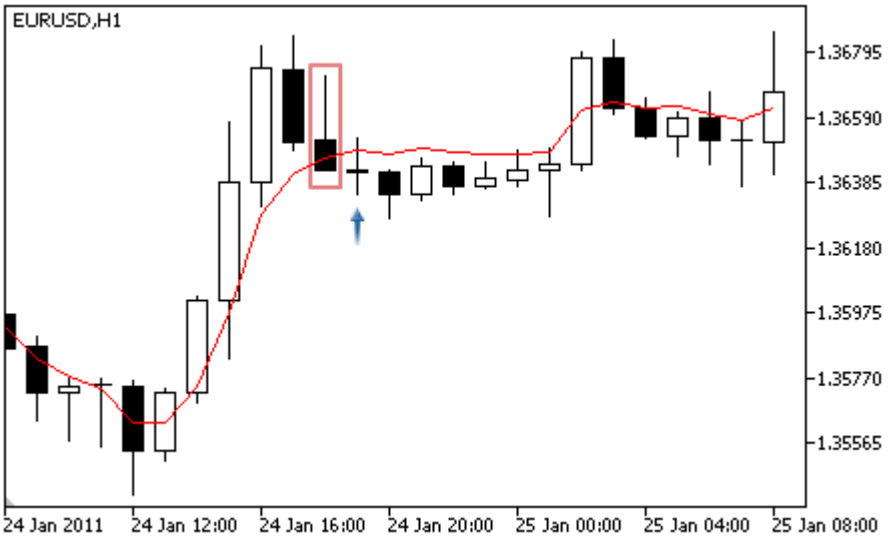
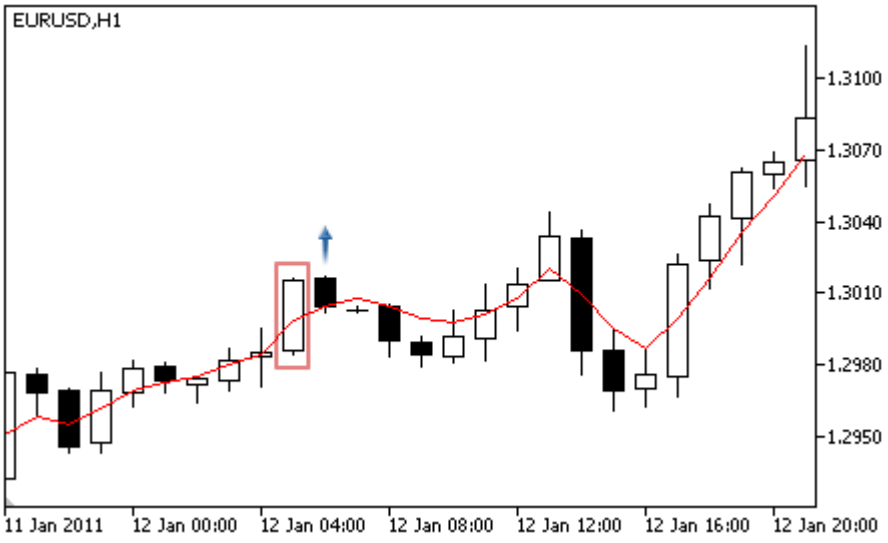
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodTriX	Periodo di calcolo dell'oscillatore.
Applied	Una serie di prezzi utilizzata per il calcolo dell'oscillatore.

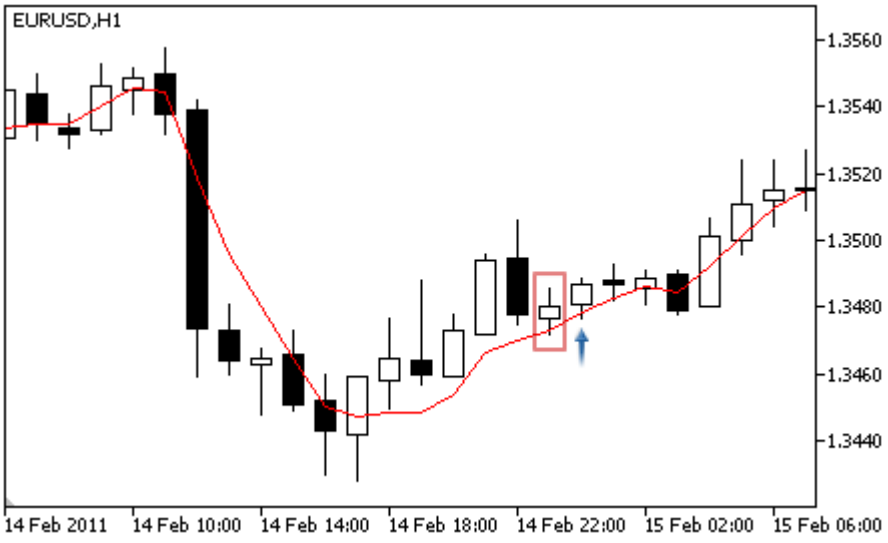
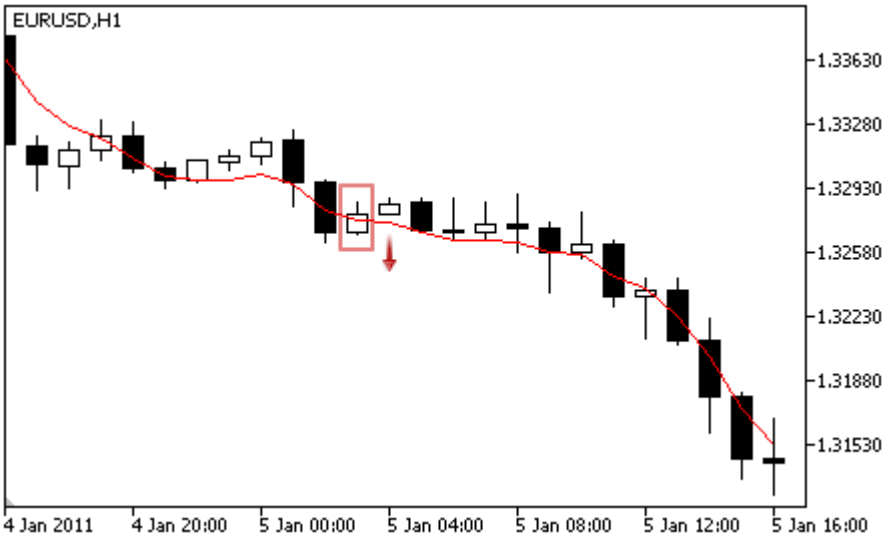
I segnali dell' Indicatore Triple Exponential Moving Average

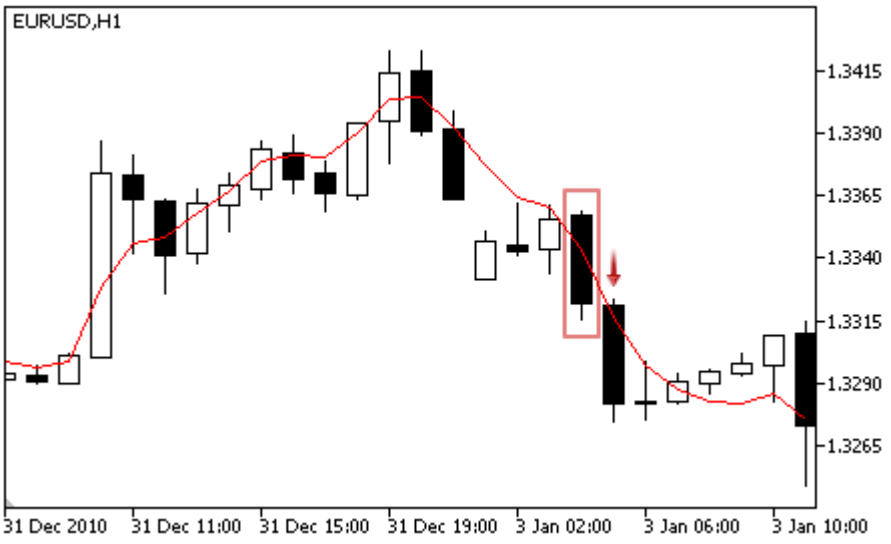
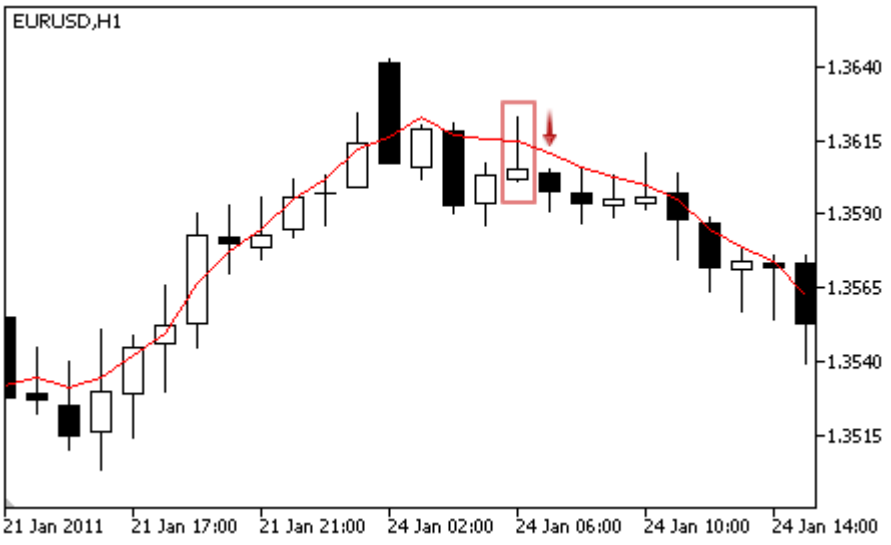
Questo modulo è dei segnali sulla base dei modelli di mercato dell'indicatore Triple Exponential Moving Average. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• Il prezzo ha incrociato l'indicatore verso il basso (il prezzo di Apertura della barra analizzata è sopra l'indicatore e il prezzo di Chiusura è sotto l'indicatore) e l'indicatore sale (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di apertura della barra analizzata è sotto l'indicatore ed il prezzo di chiusura è sopra l'indicatore) e l'indicatore sale (segnale forte). 

Tipo di segnale	Descrizione delle condizioni
	• L'ombra inferiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono al di sopra dell'indicatore, ed il prezzo Basso(Low) è sotto l'indicatore) e l'indicatore aumenta (segnale debole). 
Per il selling	• Il prezzo ha attraversato l'indicatore verso l'alto (il prezzo di Apertura(Open) della barra analizzata è sotto l'indicatore ed il prezzo di Chiusura(Close) è sopra l' indicatore) e l'indicatore scende (segnale debole).  • **Moving Average Crossover.** Il prezzo ha attraversato l'indicatore verso il basso (il prezzo di apertura(Open) della barra analizzata è sopra l'indicatore ed il prezzo di chiusura(Close) è sotto l'indicatore) e l'indicatore scende (segnale forte).

Tipo di segnale	Descrizione delle condizioni
	 • L'ombra superiore della barra ha attraversato l'indicatore (i prezzi di Apertura(Open) e Chiusura(Close) della barra analizzata sono sotto l'indicatore, ed il prezzo High è al di sopra dell' indicatore) e l'indicatore scende (segnale debole). 
Nessuna obiezione per il buying	Il prezzo è sopra l'indicatore.
Nessuna obiezione alla vendita	Il prezzo è sotto l'indicatore.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

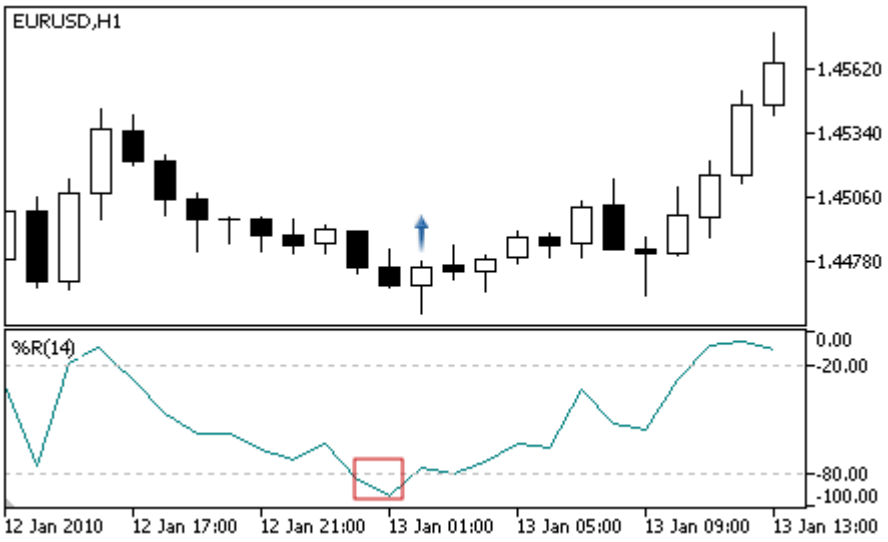
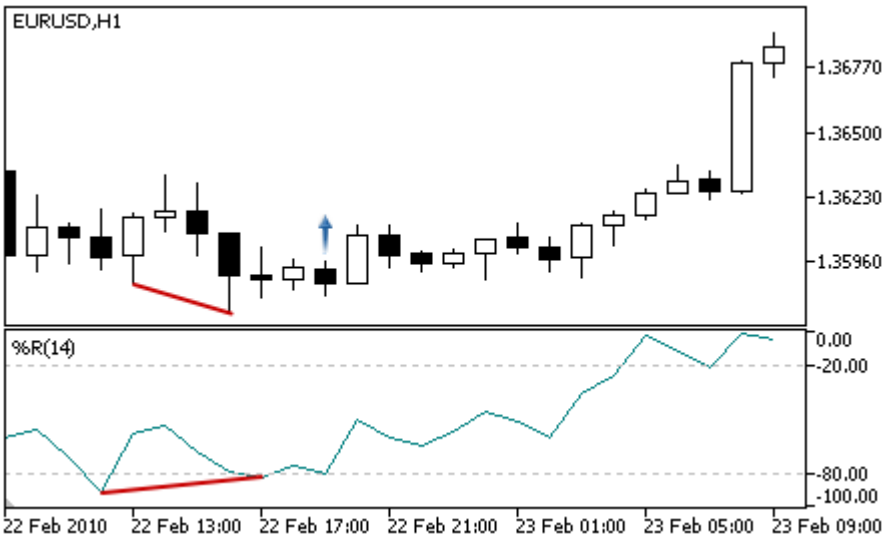
Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodMA	Periodo di media dell'indicatore.
Shift	Slittamento dell'indicatore lungo l'asse temporale (in barre).
Metodo	Metodo di calcolo della media.
Applied	A price series usate per il calcolo dell'indicatore.

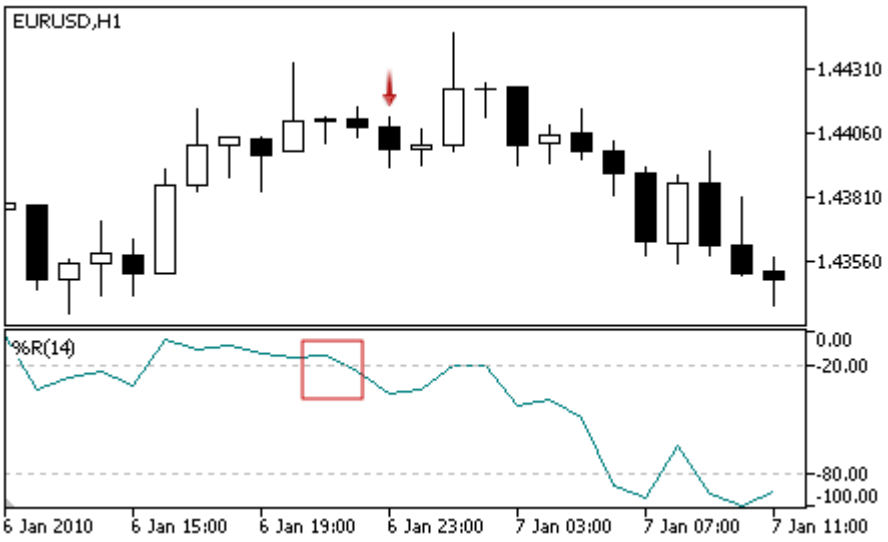
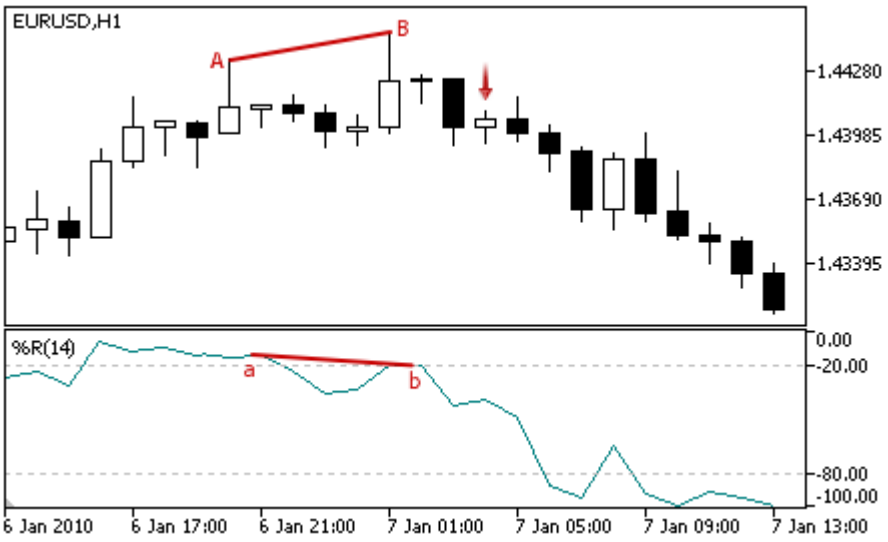
Segnali dell'oscillatore Williams Percent Range

Questo modulo di segnali si basa sui modelli di mercato dell'oscillatore Williams Percent Range. Il meccanismo di presa di decisioni di trading sulla base di segnali ottenuti dai moduli, è descritto in una sezione separata.

Condizioni di Generazione di Segnali

Qui di seguito potete trovare la descrizione delle condizioni quando il modulo passa un segnale ad un Expert Advisor.

Tipo di segnale	Descrizione delle condizioni
Per il buying	• **Reverse dietro il livello di oversold** — l'oscillatore si rivolta verso l'alto ed il suo valore alla barra analizzata è dietro il livello di ipervenduto (valore di default è -20).  • **Divergenza** — il primo fondo analizzato dell'oscillatore è superiore a quello precedente, ed il corrispondente fondo(in basso) prezzo è inferiore al precedente. 

Tipo di segnale	Descrizione delle condizioni
Per il selling	• **Reverse dietro il livello di ipercomprato** — l'oscillatore rivolto verso il basso e il suo valore al barra analizzata è dietro il livello di ipercomprato (valore di default è -80).  • **Divergenza** — il primo picco analizzato dell'oscillatore è inferiore a quello precedente ed il prezzo picco corrispondente è superiore al picco precedente. 
Nessuna obiezione per il buying	Valore dell'oscillatore cresce alla bar analizzata.
Nessuna obiezione alla vendita	Valore dell'oscillatore cade alla barra analizzata.

Nota

A seconda della modalità di funzionamento di un Expert Advisor ("Ogni tick" o "Solo prezzi di Apertura") una barra analizzata è o la barra corrente (con indice 0), o l'ultima barra formata (con indice 1).

Ricordate che l'oscillatore Williams Percent Range ha una scala rovesciata. Il suo valore massimo è -100, e il minimo è 0.

I Parametri Regolabili

Questo modulo ha i seguenti parametri regolabili:

Parametro	Descrizione
Weight	Peso del segnale del modulo nell'intervallo da 0 a 1.
PeriodWPR	Periodo di calcolo dell'oscillatore.

Trailing Stop classes

Questa sezione contiene i dettagli tecnici di lavoro con classi di trailing stop e la descrizione dei componenti rilevanti della libreria standard MQL5.

L'uso di queste classi vi farà risparmiare tempo durante la creazione (e test) di strategie di trading.

La MQL5 Standard Library (in termini di strategie di trading) si trova nella directory del terminale, nella cartella Include\Expert\Trailing.

Classe	Descrizione
CTrailingFixedPips	Questa classe implementa l'algoritmo di Trailing Stop sulla base di punti fissi
CTrailingMA	Questa classe implementa l'algoritmo Trailing Stop in base ai valori dell' indicatore Moving Average
CTrailingNone	Una classe stub, non utilizza nessun algoritmo di Trailing Stop
CTrailingPSAR	Questa classe implementa l'algoritmo Trailing Stop in base ai valori dell' indicatore Parabolic SAR

CTrailingFixedPips

CTrailingFixedPips è una classe con l'attuazione di algoritmo Trailing Stop sulla base di trailing di punti fissi .

Se la posizione ha il prezzo di Stop Loss, controlla la distanza Stop Loss minima ammessa al prezzo corrente. Se il suo valore è inferiore al livello di Stop Loss, suggerisce di fissare un nuovo prezzo di Stop Loss. Per questo caso, se la posizione ha il prezzo Take Profit, suggerisce di impostare nuovo prezzo di Take Profit.

Se Expert Advisor è stato inizializzato con il every_tick flag=false, esso deve svolgere tutte le operazioni (trading, trailing, ecc) solo alla nuova barra. Per questo caso, il livello di take profit può essere utilizzato. Essa vi permetterà di chiudere la posizione aperta al prezzo di Take Profit prima che la nuova barra sarà completata.

Descrizione

CTrailingFixedPips implementa l'algoritmo Trailing Stop sulla base di posizioni di trailing con i punti fissi.

Dichiarazione

```
class CTrailingFixedPips: public CExpertTrailing
```

Titolo

```
   #include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertTrailing
      CTrailingFixedPips

Metodi della Classe

Inizializzazione	
StopLevel	Imposta il valore del livello di Stop Loss
ProfitLevel	Imposta il valore del livello di Take Profit
virtual ValidationSettings	Controlla le impostazioni
Controlli Metodi di Trailing	
virtual CheckTrailingStopLong	Verificare le condizioni Trailing Stop della posizione long
virtual CheckTrailingStopShort	Controlla condizioni di Trailing Stop della posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

StopLevel

Imposta il valore del livello di Stop Loss (in punti).

```
void  StopLevel(
   int  stop_level      // livello Stop Loss
   )
```

Parametri

stop_loss

[in] Il valore del livello di Stop Loss (in convenzionali punti cifre 2/4).

Nota

Se il livello di Stop Loss è uguale a 0, il Trailing Stop non viene utilizzato.

ProfitLevel

Imposta il valore del livello di Take Profit (in punti).

```
void  ProfitLevel(
   int  profit_level      // livello Take profit
   )
```

Parametri

profit_level

[inlll valore del livello di Take Profit (in convenzionali 2/4 punti-cifre).

Nota

Se il profitto è uguale a 0, il Trailing Stop non viene utilizzato.

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

La funzione controlla i livelli di Take Profit e Stop Loss. I valori corretti sono 0 e valori superiori del rientro minimo in punti dal prezzo close corrente per effettuare ordini di Stop.

CheckTrailingStopLong

Controlla condizioni di Trailing Stop della posizione long.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // puntatore oggetto CPositionInfo
   double&         sl,           // prezzo Stop Loss
   double&         tp            // prezzo Take Profit
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

Se il livello di Stop Loss è uguale a 0, il Trailing Stop non viene utilizzato. Se la posizione ha già il prezzo di Stop Loss, il suo valore viene assunto come prezzo base, altrimenti il prezzo di apertura della posizione viene assunto come prezzo base.

Se il prezzo del prezzo Bid corrente è superiore al livello base di prezzo è stop loss, suggerisce di impostare il nuovo prezzo di Stop Loss. In questo caso, se la posizione ha il prezzo Take Profit, suggerisce di impostare nuovo prezzo di Take Profit.

CheckTrailingStopShort

Controlla condizioni di Trailing Stop della posizione short

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // puntatore oggetto CPositionInfo
   double&         sl,           // prezzo Stop Loss
   double&         tp            // prezzo Take Profit
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

Se il livello di Stop Loss è uguale a 0, il Trailing Stop non viene utilizzato. Se la posizione ha già il prezzo di Stop Loss, il suo valore viene assunto come prezzo base, altrimenti il prezzo di apertura della posizione viene assunto come prezzo base.

Se il corrente prezzo Ask è inferiore a livello di prezzo base - livello stop loss, suggerisce di impostare il nuovo prezzo di Stop Loss. In questo caso, se la posizione ha il prezzo Take Profit, suggerisce di impostare nuovo prezzo di Take Profit. uguale al prezzo Ask - livello take profit.

CTrailingMA

CTrailingMA è una classe di attuazione dell'algoritmo Trailing Stop basato sui valori dell'indicatore media mobile.

Descrizione

La classe CTrailingMA implementa l' algoritmo Trailing Stop in base ai valori dell'indicatore media mobile della barra precedente (completata).

Dichiarazione

```
class CTrailingMA: public CExpertTrailing
```

Titolo

```
   #include <Expert\Trailing\TrailingMA.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertTrailing
      CTrailingMA

Metodi della Classe

Inizializzazione	
Period	Imposta periodo di media mobile
Shift	Imposta slittamento di media mobile
Metodo	Imposta il metodo di smussamento della media mobile
Applied	Imposta il prezzo applicato della media mobile
virtual InitIndicators	Inizializza indicatori e serie temporali
virtual ValidationSettings	Controlla le impostazioni
Controlli Metodi di Trailing	
virtual CheckTrailingStopLong	Verificare le condizioni Trailing Stop della posizione long
virtual CheckTrailingStopShort	Controlla condizioni di Trailing Stop della posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

Period

Imposta periodo di media mobile.

```
void  Period(
   int  period      // periodo di smussamento(Smoothing period)
   )
```

Parametri

period

[in] Periodo di media mobile.

Shift

Imposta lo slittamento della media mobile.

```
void  Shift(
   int  shift      // Slittamento
   )
```

Parametri

shift

[in] Slittamento della media mobile.

Metodo

Imposta metodo di smussamento della media mobile.

```
void  Method(
   ENUM_MA_METHOD  method      // metodo di smussamento (Smoothing method)
   )
```

Parametri

method

[in] Metodo di smussamento dell'indicatore media mobile.

Applied

Imposta prezzo applicato della media mobile.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // Prezzo applicato
   )
```

Parametri

applied

[in] Prezzo Applicato della media mobile.

InitIndicators

Inizializza indicatori e serie storiche.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators puntatore collezione
   )
```

Parametri

indicators

[in] Puntatore alle collezioni indicatori e raccolta serie storiche (CExpert membro della classe).

Valore di ritorno

true in caso di successo, altrimenti false.

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

La funzione controlla il periodo di media mobile, i valori corretti sono positivi.

CheckTrailingStopLong

Controlla condizioni di Trailing Stop della posizione long.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // puntatore oggetto CPositionInfo
   double&         sl,           // prezzo Stop Loss
   double&         tp            // prezzo Take Profit
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

In primo luogo si calcola il prezzo di Stop Loss massimo consentito più vicino al prezzo corrente e calcola prezzo di Stop Loss utilizzando i valori dell'indicatore moving average della barra precedente (completata).

Se la posizione ha già prezzo di Stop Loss, il suo valore viene assunto come prezzo base, altrimenti il prezzo base è il prezzo di apertura della posizione.

Se il prezzo di Stop Loss calcolato è superiore al prezzo base ed inferiore al massimo consentito prezzo di Stop Loss, suggerisce di impostare il nuovo prezzo di Stop Loss.

CheckTrailingStopShort

Controlla condizioni di Trailing Stop della posizione short

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // puntatore oggetto CPositionInfo
   double&         sl,           // prezzo Stop Loss
   double&         tp            // prezzo Take Profit
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

In primo luogo si calcola il prezzo di Stop Loss minimo consentito più vicino al prezzo corrente e calcola prezzo di Stop Loss utilizzando i valori dell'indicatore moving average della barra precedente (completata).

Se la posizione ha già prezzo di Stop Loss, il suo valore viene assunto come prezzo base, altrimenti il prezzo base è il prezzo di apertura della posizione.

Se il prezzo di Stop Loss calcolato è superiore al prezzo base ed inferiore al minimo consentito prezzo di Stop Loss, suggerisce di impostare il nuovo prezzo di Stop Loss.

CTrailingNone

CTrailingNone una classetronca. Questa classe deve essere utilizzata in fase di inizializzazione dell'oggetto Trailng se la vostra strategia non usa il Trailing Stop.

Descrizione

La Classe CTrailingNone non implementa nessun algoritmo di Trailing Stop. I metodi di verifica delle condizioni Trailing Stop restituiscono sempre false.

Dichiarazione

```
class CTrailingNone: public CExpertTrailing
```

Titolo

```
   #include <Expert\Trailing\TrailingNone.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

Metodi della Classe

Controlli Metodi di Trailing	
virtual CheckTrailingStopLong	Un metodo stub per il check della condizioni Trailing Stop della posizione long
virtual CheckTrailingStopShort	Un metodo stub per il check della condizioni Trailing Stop della posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

Metodi ereditati dalla classe CExpertTrailing

CheckTrailingStopLong, CheckTrailingStopShort

CheckTrailingStopLong

Controlla condizioni di Trailing Stop della posizione long.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // puntatore oggetto CPositionInfo
   double&         sl,           // prezzo Stop Loss
   double&         tp            // prezzo Take Profit
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

La funzione restituisce sempre false.

CheckTrailingStopShort

Controlla condizioni di Trailing Stop della posizione short

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // puntatore oggetto CPositionInfo
   double&         sl,           // prezzo Stop Loss
   double&         tp            // prezzo Take Profit
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

La funzione restituisce sempre false.

CTrailingPSAR

CTrailingPSAR è una classe di attuazione dell'algotirmo di Trailing Stop basato sui valori dell' indicatore Parabolic SAR.

Descrizione

La classe CTrailingPSAR implementa l'algoritmo Trailing Stop in base ai valori dell' indicatore Parabolic SAR della barra precedente (completata).

Dichiarazione

```
class CTrailingPSAR: public CExpertTrailing
```

Titolo

```
   #include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

Metodi della Classe

Inizializzazione	
Step	Imposta il valore del passo dell' indicatore Parabolic SAR
Maximum	Imposta il valore di massimo dell' indicatore Parabolic SAR
virtual InitIndicators	Inizializza indicatori e serie temporali
Controlli Metodi di Trailing	
virtual CheckTrailingStopLong	Verifica le condizioni di trailing stop della posizione long
virtual CheckTrailingStopShort	Verifica le condizioni di trailing stop della posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

Step

Imposta il valore di step dell' indicatore Parabolic SAR.

```
void  Step(
   double  step      // Step
   )
```

Parametri

step

[In] Il valore di step dell' indicatore Parabolic SAR.

Maximum

Imposta il valore di massimo dell' indicatore Parabolic SAR.

```
void  Maximum(
   double  maximum      // Massimo
   )
```

Parametri

maximum

[in] Il valore massimo dell' indicatore Parabolic SAR.

InitIndicators

Inizializza indicatori e serie storiche.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators puntatore collezione
   )
```

Parametri

indicators

[in] Puntatore alle collezioni indicatori e raccolta serie storiche (CExpert membro della classe).

Valore di ritorno

true in caso di successo, altrimenti false.

CheckTrailingStopLong

Controlla condizioni di Trailing Stop della posizione long.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // Puntatore
   double&         sl,            // Link
   double&         tp             // Link
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

In primo luogo si calcola il prezzo di Stop Loss massimo consentito più vicino al prezzo corrente e calcola prezzo di Stop Loss utilizzando i valori dell'indicatore Parabolic SAR della barra precedente (completata).

Se la posizione ha già il prezzo di Stop Loss, il suo valore viene assunto come prezzo base, altrimenti il prezzo di apertura della posizione viene assunto come prezzo base.

Se il prezzo di Stop Loss calcolato è superiore al prezzo base ed inferiore al massimo consentito prezzo di Stop Loss, suggerisce di impostare il nuovo prezzo di Stop Loss.

CheckTrailingStopShort

Controlla condizioni di Trailing Stop della posizione short

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // Puntatore
   double&         sl,           // Link
   double&         tp            // Link
   )
```

Parametri

position

[In] Puntatore all'oggetto CPositionInfo.

sl

[in][out] Variabile per prezzo Stop Loss.

tp

[in][out] Variabile per il prezzo Take Profit.

Valore di ritorno

true se vengono soddisfatte le condizioni, altrimenti false.

Nota

In primo luogo si calcola il prezzo di Stop Loss minimo consentito più vicino al prezzo corrente e calcola prezzo di Stop Loss utilizzando i valori dell'indicatore Parabolic SAR della barra precedente (completata).

Se la posizione ha già il prezzo di Stop Loss, il suo valore viene assunto come prezzo base, altrimenti il prezzo di apertura della posizione viene assunto come prezzo base.

Se il prezzo di Stop Loss calcolato è superiore al prezzo base ed inferiore al minimo consentito prezzo di Stop Loss, suggerisce di impostare il nuovo prezzo di Stop Loss.

Classi Money Management

Questa sezione contiene i dettagli tecnici di lavoro con i soldi e le classi di gestione del rischio e la descrizione dei componenti rilevanti della libreria standard MQL5.

L'uso di queste classi vi farà risparmiare tempo durante la creazione (e test) di strategie di trading.

La MQL5 standard Library (in termini di classi riguardo il denaro e gestione del rischio) viene inserita nella directory del terminale, nella cartella Include\Expert\Money\.

Classe	Descrizione
CMoneyFixedLot	Questa classe implementa un algoritmo di gestione del denaro sulla base del trading con la dimensione del lotto fissa predefinita.
CMoneyFixedMargin	Questa classe implementa un algoritmo di gestione del denaro sulla base del trading con margine fisso predefinito.
CMoneyFixedRisk	Questa classe implementa un algoritmo di gestione del denaro sulla base diel trading con il rischio predefinito.
CMoneyNone	Questa classe implementa un algoritmo di gestione del denaro sulla base del trading con la dimensione del lotto minima ammessa.
CMoneySizeOptimized	Questa classe implementa un algoritmo di gestione del denaro sulla base del trading con dimensioni del lotto variabili, a seconda dei risultati dei precedenti affari.

CMoneyFixedLot

CMoneyFixedLot è una classe progettata per implementare l'algoritmo di gestione del denaro basata sul trading con predefinita la dimensione del lotto fissa.

Descrizione

CMoneyFixedLot implementa un algoritmo di gestione del denaro sulla base del trading con predefinito la dimensione del lotto fissa.

Dichiarazione

```
class CMoneyFixedLot: public CExpertMoney
```

Titolo

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

Metodi della Classe

Inizializzazione	
Lots	Imposta il volume di trading
virtual ValidationSettings	Controlla le impostazioni
Metodi di Gestione del Rischio e del Denaro	
virtual CheckOpenLong	Ottiene il volume di trading per la posizione long
virtual CheckOpenShort	Ottiene il volume di trading per la posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Metodi ereditati dalla classe CExpertMoney

Percent, CheckReverse, CheckClose

Lots

Impostazione volume di trading (in lotti).

```
void  Lots(
   double  lots      // Lotti
   )
```

Parametri

lots

[in] Volume di trading (in lotti).

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Controlla per correttezza il volume di trading specificato.

CheckOpenLong

Ottiene il volume degli scambi per la posizione long.

```
virtual double  CheckOpenLong(
   double  price,     // Prezzo
   double  sl         // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trading per la posizione long.

Nota

La funzione restituisce sempre il volume di trading fisso, definito dal metodo Lots.

CheckOpenShort

Ottiene il volume di trading per la posizione short.

```
virtual double  CheckOpenShort(
   double  price,     // Prezzo
   double  sl         // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trade per la posizione short.

Nota

La funzione restituisce sempre il volume di trading fisso, definito dal metodo Lots.

CMoneyFixedMargin

CMoneyFixedMargin è la classe progettata per implementare l'algoritmo di gestione del denaro sulla base del trading con margine fisso predefinito.

Descrizione

CMoneyFixedMargin implementa un algoritmo di gestione del denaro sulla base del trading con margine fisso predefinito.

Dichiarazione

```
class CMoneyFixedMargin: public CExpertMoney
```

Titolo

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedMargin

Metodi della Classe

Metodi di Gestione del Rischio e del Denaro	
virtual CheckOpenLong	Ottiene il volume di trading per la posizione long
virtual CheckOpenShort	Ottiene il volume di trading per la posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Metodi ereditati dalla classe CExpertMoney

Percent, ValidationSettings, CheckReverse, CheckClose

CheckOpenLong

Ottiene il volume degli scambi per la posizione long.

```
virtual double  CheckOpenLong(
   double  price,      // Prezzo
   double  sl          // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trading per la posizione long.

Nota

La funzione restituisce volume di trade per la posizione long, utilizza il margine fisso. Il margine è definito dal parametro Percentuale della classe base CExpertMoney.

CheckOpenShort

Ottiene il volume di trade per la posizione short.

```
virtual double  CheckOpenShort(
   double  price,     // Prezzo
   double  sl         // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trade per la posizione short.

Nota

La funzione restituisce il volume di trade per la posizione short, utilizza il margine fisso. Il margine è definito dal parametro Percentuale della classe base CExpertMoney.

CMoneyFixedRisk

CMoneyFixedRisk è una classe con l'attuazione dell' algoritmo gestione del denaro con il rischio fisso predefinito.

Descrizione

Classe CMoneyFixedRisk implementa l'algoritmo di gestione del denaro con il rischio predefinito fisso.

Dichiarazione

```
class CMoneyFixedRisk: public CExpertMoney
```

Titolo

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedRisk

Metodi della Classe

Metodi di Gestione del Rischio e del Denaro	
virtual CheckOpenLong	Ottiene il volume di trading per la posizione long
virtual CheckOpenShort	Ottiene il volume di trading per la posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Metodi ereditati dalla classe CExpertMoney

Percent, ValidationSettings, CheckReverse

CheckOpenLong

Ottiene il volume degli scambi per la posizione long.

```
virtual double  CheckOpenLong(
   double  price,      // Prezzo
   double  sl          // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trading per la posizione long.

Nota

La funzione restituisce Il volume di trade per la posizione long; usa il rischio fisso. Il rischio è definito dal parametro Percentuale della classe base CExpertMoney.

CheckOpenShort

Ottiene il volume di trade per la posizione short.

```
virtual double  CheckOpenShort(
   double  price,      // Prezzo
   double  sl          // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trade per la posizione short.

Nota

La funzione restituisce il volume di trade per la posizione short; usa il rischio fisso. Il rischio è definito dal parametro Percentuale della classe base CExpertMoney.

CMoneyNone

CMoneyNone è una classe con l'implementazione dell' algoritmo di trading con il lotto minimo consentito.

Descrizione

La Classe CMoneyNone implementa il trading con il lotto minimo permesso.

Dichiarazione

```
class CMoneyNone: public CExpertMoney
```

Titolo

```
#include <Expert\Money\MoneyNone.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

Metodi della Classe

Inizializzazione	
virtual ValidationSettings	Controlla le impostazioni
Metodi di Gestione del Rischio e del Denaro	
virtual CheckOpenLong	Ottiene il volume di trading per la posizione long
virtual CheckOpenShort	Ottiene il volume di trading per la posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Metodi ereditati dalla classe CExpertMoney

Percent, CheckReverse, CheckClose

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

La funzione restituisce sempre true.

CheckOpenLong

Ottiene il volume degli scambi per la posizione long.

```
virtual double  CheckOpenLong(
   double  price,     // Prezzo
   double  sl         // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trading per la posizione long.

Nota

La funzione restituisce sempre la dimensione minima del lotto.

CheckOpenShort

Ottiene il volume degli scambi per la posizione long.

```
virtual double  CheckOpenShort(
   double  price,     // Prezzo
   double  sl         // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trade per la posizione short.

Nota

La funzione restituisce sempre la dimensione minima del lotto.

CMoneySizeOptimized

CMoneySizeOptimized è una classe con implementazione di algoritmo di gestione del denaro sulla base del trading con dimensioni del lotto variabili, a seconda dei risultati dei precedenti affari(deals).

Descrizione

CMoneySizeOptimized implementa un algoritmo di gestione del denaro sulla base del trading con dimensioni del lotto variabili, a seconda dei risultati dei precedenti deals.

Dichiarazione

```
class CMoneySizeOptimized: public CExpertMoney
```

Titolo

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Gerarchia di ereditarietà

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

Metodi della Classe

Inizializzazione	
DecreaseFactor	Imposta il valore del fattore di riduzione
virtual ValidationSettings	Controlla le impostazioni
Metodi di Gestione del Rischio e del Denaro	
virtual CheckOpenLong	Ottiene il volume di trading per la posizione long
virtual CheckOpenShort	Ottiene il volume di trading per la posizione short

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Metodi ereditati dalla classe CExpertMoney

Percent, CheckReverse, CheckClose

DecreaseFactor

Imposta il valore del fattore di diminuzione.

```
void  DecreaseFactor(
   double  decrease_factor      // Fattore di diminuzione(decremento)
   )
```

Parametri

decrease_factor

[in] Fattore decremento.

Nota

DecreaseFactor definisce il coefficiente del volume decrescente (rispetto al volume della posizione precedente) per il caso di perdita consecutiva di trades.

ValidationSettings

Imposta lo slittamento.

```
virtual bool  ValidationSettings()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Se il valore del fattore di decremento è negativo, restituisce false, altrimenti restituisce true.

CheckOpenLong

Ottiene il volume degli scambi per la posizione long.

```
virtual double  CheckOpenLong(
   double  price,     // Prezzo
   double  sl         // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trading per la posizione long.

Nota

La funzione restituisce il volume di trade per la posizione long, il volume dipende dai risultati delle precedenti offerte.

CheckOpenShort

Ottiene il volume di trading per la posizione short.

```
virtual double  CheckOpenShort(
   double  price,     // Prezzo
   double  sl         // Prezzo Stop Loss
   )
```

Parametri

price

[in] Prezzo.

sl

[in] Prezzo Stop Loss.

Valore di ritorno

il volume di trading per la posizione long.

Nota

La funzione restutuisce il volume di trade per la posizione short, il volume dipende dai risultati dei precedenti affari(deals).

Le classi per la creazione di Pannelli di Controllo e Finestre di Dialogo

Questa sezione contiene i dettagli tecnici di lavoro con le classi per la creazione di Pannelli di Controllo e la descrizione dei componenti rilevanti della libreria standard MQL5.

L'uso di queste classi vi farà risparmiare tempo durante la creazione di pannelli di controllo per programmi MQL5 (Expert Advisors ed indicatori).

La MQL5 standard Library (in termini di controlli) è posizionata nella cartella dei dati del terminale client, in MQL5\Include\Controls.

Trova gli esempi di utilizzo delle classi nei seguenti articoli:

- Come creare un pannello grafico di qualsiasi livello di complessità
- Miglioramento dei Pannelli: Aggiunta di trasparenza, modifica del colore di sfondo ed ereditazione da CAppDialog/CWndClient
- Aggiunta di un pannello di controllo ad un indicatore o ad un Expert Advisor in pochissimo tempo
- Crea i tuoi pannelli grafici in MQL5
- Creazione di pannelli di controllo attivi in MQL5 per il trading

L'esempio di Expert Advisor, che illustra il funzionamento di queste classi può essere trovato in MQL5\Expert\Examples\Controls.

Auxiliary structures	Descrizione
CRect	Struttura dell'area rettangolare
CDateTime	Struttura per lavorare con la data e l'ora

Classi Base	Descrizione
CWnd	Classe di base per tutti i controlli
CWndObj	Classe base per i controlli e le finestre di dialogo
CWndContainer	Classe base per i controlli complessi (contenente i controlli dipendenti)

Comandi semplici	Descrizione
CLabel	Controllo, basato sull'oggetto grafico "Etichetta di testo"
CBmpButton	Controllo, basato sull'oggetto grafico "Etichetta Bitmap"
CButton	Controllo, basato sull'oggetto grafico "Bottone"

Comandi semplici	Descrizione
CEdit	Controllo, basato sull'oggetto grafico "Campo di Modifica"
CPanel	Controllo, basato sull'oggetto grafico "Etichetta Rettangolo"
CPicture	Controllo, basato sull'oggetto grafico "Etichetta Bitmap"

Complex controls	Descrizione
CScroll	Classe di base della barra di scorrimento
CScrollV	Barra di scorrimento verticale
CScrollH	Barra di scorrimento orizzontale
CWndClient	Classe di base dell'area client con barre di scorrimento
CListView	ListView
CComboBox	ComboBox
CCheckBox	CheckBox
CCheckGroup	CheckGroup
CRadioButton	RadioButton
CRadioGroup	RadioGroup
CSpinEdit	SpinEdit
CDialog	Dialogo
CAppDialog	Dialog Applicazione

CRect

CRect è una classe dell'area rettangolare del chart.

Descrizione

CRect è una classe dell'area, è definita da due coordinate degli angoli superiore sinistro e inferiore destro di un rettangolo in coordinate Cartesiane.

Dichiarazione

```
class CRect
```

Titolo

```
#include <Controls\Rect.mqh>
```

Metodi della Classe

Proprietà	
Left	Ottiene/Imposta la coordinata X nell'angolo in alto a sinistra
Top	Ottiene/Imposta la coordinata Y dell'angolo in alto a sinistra
Right	Ottiene/Imposta la coordinata X nell'angolo in basso a destra
Bottom	Ottiene/Imposta la coordinata Y dell'angolo in basso a destra
Larghezza	Ottiene/Imposta la larghezza
Altezza	Ottiene/Imposta l'altezza
SetBound	Imposta le nuove coordinate dell'area utilizzando le coordinate della classe CRect
Move	Imposta nuove coordinate della classe CRect
Shift	Esegue lo slittamento relativo delle coordinate CRect
Contains	Controlla se il punto è all'interno dell'area della classe CRect
Metodi aggiuntivi	
Format	Ottiene le coordinate dell'area come stringa

Left (Metodo Get)

Ottiene la coordinata X nell'angolo in alto a sinistra.

```
int  Left()
```

Valore di ritorno

Coordinata X nell'angolo in alto a sinistra.

Left (Metodo Set)

Imposta la coordinata X nell'angolo in alto a sinistra.

```
void  Left(
   const int  x      // nuove coordinate x
   )
```

Parametri

x

[in] Nuovo coordinata X nell'angolo in alto a sinistra.

Valore di ritorno

Nessuno.

Top (Metodo Get)

Ottiene la coordinata Y dell'angolo in alto a sinistra.

```
int  Top()
```

Valore di ritorno

Coordinata Y del nell'angolo in alto a sinistra.

Top (Metodo Set)

Imposta la coordinata Y dell'angolo in alto a sinistra.

```
void  Top(
   const int  y      // y coordinate
   )
```

Parametri

y

[in] Nuova coordinata Y nell'angolo in alto a sinistra.

Valore di ritorno

Nessuno.

Right (Metodo Get)

Ottiene la coordinata X nell'angolo in basso a destra.

```
int  Right()
```

Valore di ritorno

Coordinata X nell'angolo in basso a destra.

Right (Metodo Set)

Imposta la coordinata Y dell'angolo in basso a destra.

```
void  Right(
   const int  x      // coordinate x
   )
```

Parametri

x

[in] Nuova coordinata X nell'angolo in basso a destra.

Valore di ritorno

Nessuno.

Bottom (Metoto Get)

Ottiene la coordinata Y dell'angolo in basso a destra.

```
int  Bottom()
```

Valore di ritorno

Coordinata Y dell'angolo in basso a destra.

Bottom (Metodo Set)

Imposta la coordinata Y dell'angolo in basso a destra.

```
void  Bottom(
   const int  y      // y coordinate
   )
```

Parametri

y

[in] Nuova coordinata Y dell'angolo in basso a destra.

Valore di ritorno

Nessuno.

Width (Metodo Get)

Ottiene la larghezza dell'area.

```
int  Width()
```

Valore di ritorno

Spessore dell'area.

Width (Metodo Set)

Imposta nuova larghezza dell'area.

```
virtual bool  Width(
   const int  w      // spessore
   )
```

Parametri

w

[in] Nuovo spessore.

Valore di ritorno

true in caso di successo, altrimenti false.

Height (Metoto Get)

Ottiene l'altezza dell'area.

```
int  Height()
```

Valore di ritorno

Altezza dell'area.

Height (Metodo Set)

Imposta nuova altezza dell'area.

```
virtual bool  Height(
   const int  h      // altezza
   )
```

Parametri

h

[in] Nuova altezza.

Valore di ritorno

true in caso di successo, altrimenti false.

SetBound

Imposta le nuove coordinate dell'area utilizzando le coordinate della classe CRect.

```
void  SetBound(
   const & CRect  rect      // Classe CRect
   )
```

Valore di ritorno

Nessuno.

SetBound

Imposta nuove coordinate dell'area.

```
void  SetBound(
   const int  l      // sinistra
   const int  t      // alto
   const int  r      // destra
   const int  b      // basso
   )
```

Parametri

l

[in] Coordinata X dell'angolo in alto a sinistra.

t

[in] Coordinata Y dell'angolo in alto a sinistra.

r

[in] Coordinata X dell'angolo in basso a destra.

b

[in] Coordinata Y dell'angolo in basso a destra.

Valore di ritorno

Nessuno.

Move

Imposta nuove coordinate della classe CRect.

```
void  Move(
   const int  x,     // X coordinate
   const int  y      // Y coordinate
   )
```

Parametri

x

[in] Nuovo coordinata X.

y

[in] Nuova coordinate Y.

Valore di ritorno

Nessuno.

Shift

Esegue lo slittamento relativo delle coordinate della classe CRect.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parametri

dx

[in] Delta X.

dy

[in] Delta Y.

Valore di ritorno

Nessuno.

Contains

Controlla se il punto è all'interno dell'area della classe CRect.

```
bool  Contains(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   )
```

Parametri

x

[in] X coordinate.

y

[in] Y coordinate.

Valore di ritorno

true, se il punto è all'interno dell'area (inclusi bordi), altrimenti - false.

Format

Ottiene le coordinate dell'area come stringa.

```
string  Format(
   string &  fmt,     // formato
   ) const
```

Parametri

fmt

[in] Stringa con formato.

Valore di ritorno

Stringa con le coordinate dell'area.

CDateTime

CDateTime è una struttura per lavorare con data ed ora.

Descrizione

CDateTime è una struttura derivata da MqlDateTime, è usata per le operazioni con data e ora nei controlli.

Dichiarazione

```
struct CDateTime
```

Titolo

```
#include <Tools\DateTime.mqh>
```

Metodi della Classe

Proprietà	
MonthName	Ottiene nome del mese
ShortMonthName	Ottiene nome breve del mese
DayName	Ottiene il nome completo del giorno in una settimana
ShortDayName	Ottiene breve nome del giorno in una settimana
DaysInMonth	Ottiene il numero di giorni nel mese
Medoti Get/Set	
DateTime	Metodi Gets/Sets data ed ora
Date	Imposta la data
Time	Imposta l'ora
Sec	Imposta secondi
Min	Imposta minuti
Hour	Imposta ora
Day	Imposta giorno del mese
Mon	Imposta mese
Year	Imposta l'anno
Metodi aggiuntivi	
SecDec	Sottrae un determinato numero di secondi
SecInc	Aggiunge il numero specificato di secondi
MinDec	Sottrae il numero di minuti specificato

Proprietà	
MinInc	Aggiunge il numero specificato di minuti
HourDec	Sottrae un determinato numero di ore
HourInc	Aggiunge un determinato numero di ore
DayDec	Sottrae un determinato numero di giorni
DayInc	Aggiunge un determinato numero di giorni
MonDec	Sottrae un determinato numero di mesi
MonInc	Aggiunge il numero specificato di mesi
YearDec	Sottrae un determinato numero di anni
YearInc	Aggiunge un determinato numero di anni

MonthName

Ottiene nome del mese.

```
string  MonthName()  const
```

Ottiene nome del mese in base all'indice.

```
string  MonthName(
   const int       num          // indice del mese
   )  const
```

Parametri

num

[in] Indice del mese (1-12).

Valore di ritorno

Nome del mese.

ShortMonthName

Ottiene nome breve del mese.

```
string  ShortMonthName()  const
```

Ottiene il nome breve del mese in base all'indice.

```
string  ShortMonthName(
   const int       num          // indice del mese
   )  const
```

Parametri

num

[in] Indice del mese (1-12).

Valore di ritorno

Nome breve del mese.

DayName

Ottiene il nome completo del giorno in una settimana.

```
string  DayName()  const
```

Ottiene il nome completo del giorno in una settimana in base all'indice.

```
string  DayName(
   const int      num         // indice del giorno
   )  const
```

Parametri

num

[in] Indice del giorno (0-6).

Valore di ritorno

Nome del giorno.

ShortDayName

Ottiene il breve nome del giorno in una settimana.

```
string  ShortDayName()  const
```

Ottiene breve nome del giorno in una settimana in base all'indice.

```
string  ShortDayName(
   const int      num         // indice del giorno
   )  const
```

Parametri

num

[in] Indice del giorno (0-6).

Valore di ritorno

Breve nome del giorno.

DaysInMonth

Ottiene il numero di giorni nell'ultimo mese.

```
int  DaysInMonth()  const
```

Valore di ritorno

Numero di giorni in un mese.

DateTime (Metodo Get)

Ottiene data ed ora.

```
datetime  DateTime()
```

Valore di ritorno

Valore di tipo datetime.

DateTime (Metodo Set datetime)

Imposta data e ora con il tipo datetime.

```
void  DateTime(
   const datetime      value         // data ed ora
   )
```

Parametri

value

[in] Valore di tipo datetime.

Valore di ritorno

Nessuno.

DateTime (Metodo Set MqlDateTime)

Imposta data e ora con il tipo MqlDateTime.

```
void  DateTime(
   const MqlDateTime   &value        // data ed ora
   )
```

Parametri

value

[in] Valore di tipo MqlDateTime.

Valore di ritorno

Nessuno.

Date (Metodo Set datetime)

Imposta la data con il tipo datetime.

```
void  Date(
   const datetime      value         // data
   )
```

Parametri

value

[in] Valore di tipo datetime.

Valore di ritorno

Nessuno.

Date (Set Method MqlDateTime)

Imposta la data con il tipo MqlDateTime.

```
void  Date(
   const MqlDateTime   &value        // data
   )
```

Parametri

value

[in] Valore di tipo MqlDateTime.

Valore di ritorno

Nessuno.

Time (Metodo Set datetime)

Imposta l'orario con il tipo datetime.

```
void  Time(
   const datetime      value         // orario
   )
```

Parametri

value

[in] Valore di tipo datetime.

Valore di ritorno

Nessuno.

Time (Metodo Set MqlDateTime)

Imposta l'orario con il tipo MqlDateTime.

```
void  Time(
   const MqlDateTime   &value        // orario
   )
```

Parametri

value

[in] Valore di tipo MqlDateTime.

Valore di ritorno

Nessuno.

Sec

Imposta i secondi.

```
void  Sec(
   const int   value        // secondi
   )
```

Parametri

value

[in] Secondi.

Valore di ritorno

Nessuno.

Min

Imposta minuti.

```
void  Min(
   const int    value        // minuti
   )
```

Parametri

value

[in] Minuti.

Valore di ritorno

Nessuno.

Hour

Imposta ora.

```
void  Hour(
   const int    value        // ora
   )
```

Parametri

value

[in] Hour.

Valore di ritorno

Nessuno.

Day

Imposta giorno del mese.

```
void  Day(
   const int    value        // giorno
   )
```

Parametri

value

[in] Giorno del mese.

Valore di ritorno

Nessuno.

Mon

Imposta mese.

```
void  Mon(
   const int    value        // mese
   )
```

Parametri

value

[in] Mese.

Valore di ritorno

Nessuno.

Year

Imposta l'anno.

```
void  Year(
   const int    value        // anno
   )
```

Parametri

value

[in] Anno.

Valore di ritorno

Nessuno.

SecDec

Sottrae un determinato numero di secondi.

```
void  SecDec(
   int   delta=1        // secondi
   )
```

Parametri

delta

[in] Secondi da sottrarre.

Valore di ritorno

Nessuno.

SecInc

Aggiunge il numero specificato di secondi.

```
void  SecInc(
   int   delta=1        // secondi
   )
```

Parametri

delta

[in] Secondi da sottrarre.

Valore di ritorno

Nessuno.

MinDec

Sottrae il numero di minuti specificato.

```
void  MinDec(
   int   delta=1        // minuti
   )
```

Parametri

delta

[in] Minuti da sottrarre.

Valore di ritorno

Nessuno.

MinInc

Aggiunge il numero specificato di minuti.

```
void  MinInc(
   int   delta=1        // minuti
   )
```

Parametri

delta

[in] Minuti da aggiungere.

Valore di ritorno

Nessuno.

HourDec

Sottrae un determinato numero di ore.

```
void  HourDec(
   int   delta=1        // ore
   )
```

Parametri

delta

[in] Ore da sottrarre.

Valore di ritorno

Nessuno.

HourInc

Aggiunge un determinato numero di ore.

```
void  HourInc(
   int   delta=1        // ore
   )
```

Parametri

delta

[in] Ore da aggiungere.

Valore di ritorno

Nessuno.

DayDec

Sottrae un determinato numero di giorni.

```
void  DayDec(
   int   delta=1        // giorni
   )
```

Parametri

delta

[in] Giorni da sottrarre.

Valore di ritorno

Nessuno.

DayInc

Aggiunge il numero specificato di giorni.

```
void  DayInc(
   int   delta=1        // giorni
   )
```

Parametri

delta

[in] Giorni da aggiungere.

Valore di ritorno

Nessuno.

MonDec

Sottrae un determinato numero di mesi.

```
void  MonDec(
   int   delta=1        // mesi
   )
```

Parametri

delta

[in] Mesi da sottrarre.

Valore di ritorno

Nessuno.

MonInc

Aggiunge il numero specificato di mesi

```
void  MonInc(
   int   delta=1        // mesi
   )
```

Parametri

delta

[in] Mesi da aggiungere.

Valore di ritorno

Nessuno.

YearDec

Sottrae un determinato numero di anni.

```
void  YearDec(
   int   delta=1        // anni
   )
```

Parametri

delta

[in] Anni da sottrarre.

Valore di ritorno

Nessuno.

YearInc

Aggiunge un determinato numero di anni.

```
void  YearInc(
   int   delta=1        // anni
   )
```

Parametri

delta

[in] Anni da aggiungere.

Valore di ritorno

Nessuno.

CWnd

CWnd è una classe base per tutti i controlli inclusi nella libreria standard MQL5.

Descrizione

La classe CWnd è l'implementazione della classe di controllo di base.

Dichiarazione

```
class CWnd : public CObject
```

Titolo

```
#include <Controls\Wnd.mqh>
```

Gerarchia di ereditarietà

CObject

CWnd

Discendenti diretti

CDragWnd, CWndContainer, CWndObj

Metodi della Classe

Creare e distruggere	
Create	Crea il controllo
Destroy	Distrugge controllo
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
OnMouseEvent	Event handler per l'evento CHARTEVENT_MOUSE_MOVE
Name	
Name	Ottiene il nome del controllo
Accesso al contenitore	
ControlsTotal	Ottiene il numero di controlli nel contenitore
Control	Ottiene il controllo in base all'indice
ControlFind	Ottiene il controllo per ID
Geometry	
Rect	Ottiene il puntatore all'oggetto classe CRect
Left	Ottiene/Imposta la coordinata X dell'angolo superiore sinistro

Creare e distruggere	
Top	Ottiene/Imposta la coordinata Y dell'angolo superiore sinistro
Right	Ottiene/Imposta la coordinata X dell'angolo inferiore destro
Bottom	Ottiene/Imposta la coordinata Y dell'angolo inferiore destro
Larghezza	Ottiene/Imposta la larghezza
Altezza	Ottiene/Imposta l'altezza
Move	Imposta nuove coordinate del controllo
Shift	Esegue il relativo spostamento delle coordinate di controllo
Ridimensiona	Imposta nuova larghezza/altezza del controllo
Contains	Controlla se il punto/controllo è all'interno della zona di controllo
Align	
Alignment	Imposta le proprietà di allineamento del controllo
Align	Esegue l'allineamento del controllo
Identification	
Id	Ottiene/Imposta l'ID di controllo
State	
IsEnabled	Ottiene un valore che indica se il controllo è abilitato
Enable	Imposta un valore che indica se il controllo è abilitato
Disable	Disabilita il controllo
IsVisible	Controlla la flag di visibilità
Visible	Imposta la flag di visibilità
Show	Mostra il controllo
Hide	Nasconde il controllo
IsActive	Controlla l'attività di controllo
Activate	Attiva il controllo
Deactivate	Disattiva il controllo
State flags	
StateFlags	Ottiene/Imposta le flags dello stato del controllo
StateFlagsSet	Imposta le flags dello stato del controllo
StateFlagsReset	Ripristina le flags dello stato del controllo
Properties flags	

Creare e distruggere	
PropFlags	Ottiene/imposta le flags delle proprietà del controllo
PropFlagsSet	Imposta le flags delle proprietà del controllo
PropFlagsReset	Ripristina le flags delle proprietà del controllo
Operazioni del mouse	
MouseX	Ottiene/Salva la coordinata X del mouse
MouseY	Ottiene/Salva la coordinata Y del mouse
MouseFlags	Ottiene/Salva lo stato dei pulsanti del mouse
MouseFocusKill	Elimina il focus del mouse
Event handler interni	
OnCreate	Event handler "Crea"
OnDestroy	Event handler "Distrugge"
OnMove	Event handler "Sposta"
OnResize	Event handler "Ridimensiona"
OnEnable	Event handler "Abilita"
OnDisable	Event handler "Disabilita"
OnShow	Event handler "Mostra"
OnHide	Event handler "Nascondi"
OnActivate	Event handler "Attiva"
OnDeactivate	Event handler "Disattiva"
OnClick	Event handler "Click"
OnChange	Event handler "Cambia"
Event handlers del mouse	
OnMouseDown	Event handler "Mouse giù"
OnMouseUp	Event handler "Mouse sù"
Event handlers trascinamento	
OnDragStart	Event handler "DragStart"
OnDragProcess	Event handler "DragProcess"
OnDragEnd	Event handler "DragEnd"
Drag object	
DragObjectCreate	Crea oggetto trascinamento

Creare e distruggere	
DragObjectDestroy	Distrugge oggetto trascinamento

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Create

Crea un controllo.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] Coordinata X dell'angolo in alto a sinistra.

y1

[in] Coordinata Y dell'angolo in alto a sinistra.

x2

[in] Coordinata X dell'angolo in basso a destra.

y2

[in] Coordinata Y dell'angolo in basso a destra.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Il metodo della classe base salva solo i parametri e restituisce sempre true.

Destroy

Distrugge un controllo.

```
virtual bool  Destroy()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] parametro evento di tipo long, passato per riferimento.

dparam

[in] parametro evento di tipo double, passato per riferimento.

sparam

[in] parametro evento di tipo string, passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

OnMouseEvent

Event handler del mouse (l'evento chart CHARTEVENT_MOUSE_MOVE).

```
virtual bool  OnMouseEvent(
   const int  x,          // x coordinate
   const int  y,          // y coordinate
   const int  flags       // flags
   )
```

Parametri

x

[in] coordinata X del cursore del mouse rispetto all'angolo in alto a sinistra del grafico.

y

[in] coordinata Y del cursore del mouse rispetto all'angolo in alto a sinistra del grafico.

flags

[in] Flag degli stati dei bottoni del mouse.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

Name

Ottiene il controllo nome.

```
string  Name()  const
```

Valore di ritorno

Nome del controllo.

ControlsTotal

Ottiene il numero di controlli nel contenitore.

```
int  ControlsTotal()  const
```

Valore di ritorno

Numero di controlli in un contenitore.

Nota

Il metodo della classe base non ha contenitore, fornisce l'accesso al contenitore per i suoi eredi e restituisce sempre 0.

Control

Ottiene il controllo per indice.

```
CWnd*  Control(
   const int  ind      // indice
   )  const
```

Parametri

ind

[in] Indice controllo.

Valore di ritorno

Un puntatore al controllo.

Nota

Il metodo della classe base non ha contenitore, fornisce l'accesso al contenitore per i suoi eredi e restituisce sempre NULL.

ControlFind

Ottiene il controllo dal contenitore per ID specificato.

```
virtual CWnd*  ControlFind(
   const long  id      // ID
   )
```

Parametri

id

[in] Identificatore del controllo da cercare.

Valore di ritorno

Puntatore al controllo.

Nota

Il metodo della classe base non ha contenitore, fornisce l'accesso al contenitore per i suoi eredi. Se la l'ID specificato corrisponde con l'ID del contenitore, restituisce un puntatore a se stesso (questo).

Rect

Ottiene il puntatore all'oggetto della classe CRect.

```
const CRect*  Rect()  const
```

Valore di ritorno

Puntatore all'oggetto della classe CRect.

Left (Metodo Get)

Ottiene la coordinata X nell'angolo in alto a sinistra del controllo.

```
int  Left()
```

Valore di ritorno

Coordinata X nell'angolo in alto a sinistra del controllo.

Left (Metodo Set)

Imposta la coordinata X nell'angolo in alto a sinistra del controllo.

```
void  Left(
   const int  x      // nuove coordinate x
   )
```

Parametri

x

[in] Nuovo coordinata X nell'angolo in alto a sinistra.

Valore di ritorno

Nessuno.

Top (Metodo Get)

Ottiene la coordinata Y dell'angolo superiore sinistro del controllo.

```
int  Top()
```

Valore di ritorno

Coordinata Y nell'angolo in alto a sinistra del controllo.

Top (Metodo Set)

Imposta la coordinata Y dell'angolo superiore sinistro del controllo.

```
void  Top(
   const int  y      // y coordinate
   )
```

Parametri

y

[in] Nuova coordinata Y nell'angolo in alto a sinistra.

Valore di ritorno

Nessuno.

Right (Metodo Get)

Ottiene la coordinata X nell'angolo inferiore destro del controllo.

```
int  Right()
```

Valore di ritorno

Coordinata X nell'angolo in basso a destra.

Right (Metodo Set)

Imposta la coordinata Y dell'angolo inferiore destro del controllo.

```
void  Right(
   const int  x      // coordinate x
   )
```

Parametri

x

[in] Nuova coordinata X nell'angolo in basso a destra.

Valore di ritorno

Nessuno.

Bottom (Metoto Get)

Ottiene la coordinata Y dell'angolo inferiore destro del controllo.

```
int  Bottom()
```

Valore di ritorno

Coordinata Y dell'angolo inferiore destro del controllo.

Bottom (Metodo Set)

Imposta la coordinata Y dell'angolo inferiore destro del controllo.

```
void  Bottom(
   const int  y      // y coordinate
   )
```

Parametri

y

[in] Nuova coordinata Y dell'angolo in basso a destra.

Valore di ritorno

Nessuno.

Width (Metodo Get)

Ottiene la larghezza del controllo.

```
int  Width()
```

Valore di ritorno

Sp'essore del controllo.

Width (Metodo Set)

Imposta il nuovo spessore del controllo.

```
virtual bool  Width(
   const int  w      // spessore
   )
```

Parametri

w

[in] Nuovo spessore.

Valore di ritorno

true in caso di successo, altrimenti false.

Height (Metoto Get)

Ottiene l'altezza di controllo.

```
int  Height()
```

Valore di ritorno

Altezza del controllo.

Height (Metodo Set)

Imposta nuova altezza del controllo.

```
virtual bool  Height(
   const int  h      // altezza
   )
```

Parametri

h

[in] Nuova altezza.

Valore di ritorno

true in caso di successo, altrimenti false.

Move

Imposta nuove coordinate del controllo.

```
void  Move(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   )
```

Parametri

x

[in] Nuovo coordinata X.

y

[in] Nuova coordinate Y.

Valore di ritorno

Nessuno.

Shift

Esegue lo spostamento relativo delle coordinate del controllo.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parametri

dx

[in] Delta X.

dy

[in] Delta Y.

Valore di ritorno

Nessuno.

Ridimensiona

Imposta nuova larghezza/altezza del controllo.

```
virtual bool  Resize(
   const int  w,      // larghezza
   const int  h       // altezza
   )
```

Parametri

w

[in] Nuovo spessore.

h

[in] Nuova altezza.

Valore di ritorno

true in caso di successo, altrimenti false.

Contains

Controlla se il punto è all'interno dell'area di controllo del chart.

```
bool  Contains(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   )
```

Parametri

x

[in] X coordinate.

y

[in] Y coordinate.

Valore di ritorno

true, se il punto è all'interno dell'area (inclusi bordi), altrimenti - false.

Contains

Controlla se il controllo specificato si trova all'interno dell'area di controllo del chart.

```
bool  Contains(
   const CWnd*  control      // puntatore
   )  const
```

Parametri

control

[in] Puntatore oggetto.

Valore di ritorno

true, se il controllo specificato si trova all'interno dell'area (compresi i bordi), altrimenti - false.

Alignment

Imposta parametri di allineamento del controllo.

```
void  Alignment(
   const int  flags,       // flags allineamento
   const int  left,        // offset dal bordo sinistro
   const int  top,         // offset dal bordo superiore
   const int  right,       // offset dal bordo destro
   const int  bottom       // offset dal bordo inferiore
   )
```

Parametri

flags

[in] Flags allineamento.

left

[in] Offset fissato, dal bordo sinistro.

top

[in] Offset fissato dal bordo superiore.

right

[in] Offset fissato dal bordo destro.

bottom

[in] Offset fissato dal bordo inferiore.

Valore di ritorno

Nessuno.

Nota

Flags allineamento:

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // nessun allineamento
   WND_ALIGN_LEFT=1,                                  // allinea a sinistra
   WND_ALIGN_TOP=2,                                   // allinea in alto
   WND_ALIGN_RIGHT=4,                                 // allinea a destra
   WND_ALIGN_BOTTOM=8,                                // allinea su
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // spessore allineamento
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // altezza allineamento
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // altezza e spessore allineamer
   }
```

Align

Esegue il controllo allineamento nell'area chart specificata.

```
virtual bool  Align(
   const CRect*  rect      // puntatore
   )
```

Parametri

rect

[in] Puntatore all'oggetto con coordinate area del chart.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

I parametri di allineamento devono essere specificati (nessun allineamento, di default).

Id (Metodo Get)

Ottiene l'ID del controllo.

```
long  Id()  const
```

Valore di ritorno

L'identificatore del controllo.

Id (Metodo Set)

Imposta nuovo valore dell'ID del controllo.

```
virtual long  Id(
   const long  id      // identificatore
   )
```

Parametri

id

[in] Nuovo valore dell'identificatore del controllo.

Valore di ritorno

Nessuno.

IsEnabled

Ottiene un valore che indica se il controllo è abilitato.

```
bool  IsEnabled()  const
```

Valore di ritorno

true - se il controllo è abilitato, altrimenti - false.

Enable

Abilita il controllo.

```
virtual bool  Enable()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Se il controllo è attivato, è in grado di elaborare gli eventi esterni.

Disable

Disabilita il controllo.

```
virtual bool  Disable()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Il controllo disattivato non è in grado di elaborare gli eventi esterni.

IsVisible

Ottiene un valore che indica se il controllo è visibile.

```
bool  IsVisible()  const
```

Valore di ritorno

true se il controllo viene mostrato sul grafico, altrimenti false.

Visible

Imposta il flag di visibilità.

```
virtual bool  Visible(
   const bool  flag      // flag
   )
```

Parametri

flag

[in] Nuova flag.

Valore di ritorno

true in caso di successo, altrimenti false.

Show

Mostra il controllo.

```
virtual bool  Show()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Hide

Nasconde il controllo.

```
virtual bool  Hide()
```

Valore di ritorno

true in caso di successo, altrimenti false.

IsActive

Ottiene un valore che indica se il controllo è attivo.

```
bool  IsActive()  const
```

Valore di ritorno

true se il controllo è attivo, altrimenti false.

Activate

Attiva il controllo.

```
virtual bool  Activate()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Il controllo si attiva quando il cursore del mouse passa su di esso.

Deactivate

Disattiva il controllo.

```
virtual bool  Deactivate()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Il controllo diventa inattivo quando il cursore del mouse è fuori dal controllo.

StateFlags (Metodo Get)

Ottiene le flag dello stato del controllo.

```
int  StateFlags()
```

Valore di ritorno

Le flags dello stato del controllo.

StateFlags (Metodo Set)

Imposta le flags dello Stato del controllo.

```
virtual void  StateFlags(
   const int  flags      // flags
   )
```

Parametri

flags

[in] Nuove flags dello stato del controllo.

Valore di ritorno

Nessuno.

StateFlagsSet

Imposta le flags dello Stato del controllo.

```
virtual void  StateFlagsSet(
   const int  flags      // flags
   )
```

Parametri

flags

[in] Flags da impostare (bit mask).

Valore di ritorno

Nessuno.

StateFlagsReset

Resetta le flags dello stato del controllo.

```
virtual void  StateFlagsReset(
   const int  flags      // flags
   )
```

Parametri

flags

[in] Flags da resettare (bit mask).

Valore di ritorno

Nessuno.

PropFlags (Metoto Get)

Ottiene le flag delle proprietà del controllo.

```
void  PropFlags(
   const int  flags      // flags
   )
```

Valore di ritorno

Le flag delle proprietà del controllo.

PropFlags (Metoto Set)

Imposta le flags delle proprietà del controllo.

```
virtual void  PropFlags(
   const int  flags      // flags
   )
```

Parametri

flags

[in] Nuove flags.

Valore di ritorno

Nessuno.

PropFlagsSet

Imposta le flags delle proprietà del controllo.

```
virtual void  PropFlagsSet(
   const int  flags      // flags
   )
```

Parametri

flags

[in] Flags da impostare (bit mask).

Valore di ritorno

Nessuno.

PropFlagsReset

Resetta le flags delle proprietà del controllo.

```
virtual void  PropFlagsReset(
   const int  flags      // flags
   )
```

Parametri

flags

[in] Flags da resettare (bit mask).

Valore di ritorno

Nessuno.

MouseX (Metodo Set)

Salva la coordinata del mouse X.

```
void  MouseX(
   const int  value      // coordinate
   )
```

Parametri

value

[in] La coordinata X del mouse.

Valore di ritorno

Nessuno.

MouseX (Metodo Get)

Ottiene la coordinata X salvata del mouse.

```
int  MouseX()
```

Valore di ritorno

La coordinata X salvata del mouse.

MouseY (Metodo Set)

Salva la coordinata Y del mouse.

```
void  MouseY(
   const int  value      // coordinate
   )
```

Parametri

value

[in] La coordinata Y del mouse.

Valore di ritorno

Nessuno.

MouseY (Metodo Get)

Ottiene la coordinata Y salvata del mouse.

```
int  MouseY()
```

Valore di ritorno

La coordinata Y salvata del mouse.

MouseFlags (Metodo Set)

Salva lo stato dei bottoni del mouse.

```
virtual void  MouseFlags(
   const int  value      // stato
   )
```

Parametri

value

[in] Stato dei bottoni del mouse.

Valore di ritorno

Nessuno.

MouseFlags (Metodo Get)

Ottiene lo stato salvato dei bottoni del mouse.

```
int  MouseFlags()
```

Valore di ritorno

Stato dei bottoni del mouse.

MouseFocusKill

Cancella lo stato salvato dei bottoni del mouse e disattiva il controllo.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parametri

id=CONTROLS_INVALID_ID

[in] Identificatore del controllo, che ha ricevuto il focus del mouse.

Valore di ritorno

Il risultato della disattivazione del controllo.

OnCreate

L'event handler del controllo "Create" .

```
virtual bool  OnCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnDestroy

L'event handler del controllo "Destroy" .

```
virtual bool  OnDestroy()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnMove

L'event handler del controllo "Move" .

```
virtual bool  OnMove()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnEnable

L'event handler del controllo "Enable" (se è disabilitato non risponde all'interazione dell'utente).

```
virtual bool  OnEnable()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnDisable

L'event handler del controllo "Disable" (se è disabilitato non risponde all'interazione dell'utente).

```
virtual bool  OnDisable()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnActivate

Il controllo del gestore di eventi (event handler) "Attiva"(Activate).

```
virtual bool  OnActivate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnDeactivate

Il controllo del gestore di eventi (event handler) "Disattiva"(Deactivate).

```
virtual bool  OnDeactivate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnClick

L'event handler del controllo "Click" (clic del mouse sul tasto sinistro).

```
virtual bool  OnClick()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChange

L'event handler del controllo "Change" .

```
virtual bool  OnChange()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnMouseDown

L'event handler del controllo "MouseDown" .

```
virtual bool  OnMouseDown()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "MouseDown" si verifica quando si preme il pulsante sinistro del mouse sul controllo.

OnMouseUp

Il gestore dell'evento (event handler) "MouseUp" (rilascio del pulsante sinistro del mouse) .

```
virtual bool  OnMouseUp()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "MouseDown" si verifica quando si rilascia il pulsante sinistro del mouse sul controllo.

OnDragStart

L'event handler del controllo "DragStart" .

```
virtual bool  OnDragStart()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "DragStart" si verifica all'inizio di un'operazione di trascinamento.

OnDragProcess

L'event handler del controllo "DragProcess" .

```
virtual bool  OnDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parametri

x

[in] Attuale coordinata X del cursore del mouse.

y

[in] Attuale coordinata Y del cursore del mouse.

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "DragProcess" si verifica quando il controllo viene spostato.

OnDragEnd

L'event handler del controllo "DragEnd" .

```
virtual bool  OnDragEnd()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "DragEnd" si verifica quando il processo di trascinamento del controllo è terminato.

DragObjectCreate

Crea oggetto di trascinamento.

```
virtual bool  DragObjectCreate()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

true se l'evento è stato elaborato, altrimenti false.

DragObjectDestroy

Distrugge oggetto di trascinamento.

```
virtual bool  DragObjectDestroy()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CWndObj

CWndObj è una classe base per i controlli semplici (basate su oggetti chart) della Libreria Standard.

Descrizione

La Classe CWndObj implementa metodi di base del controllo semplice.

Dichiarazione

```
class CWndObj : public CWnd
```

Titolo

```
#include <Controls\WndObj.mqh>
```

Gerarchia di ereditarietà

CObject
  CWnd
    CWndObj

Discendenti diretti

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

Metodi della Classe

Elaborazione eventi del Chart	
OnEvent	Event Handler di tutti gli eventi del chart
Proprietà	
Testo	Ottiene/Imposta la proprietà dell'oggetto chart OBJPROP_TEXT
Color	Ottiene/Imposta la proprietà dell'oggetto chart OBJPROP_COLOR
ColorBackground	Ottiene/Imposta la proprietà dell'oggetto chart OBJPROP_BGCOLOR
ColorBorder	Ottiene/Imposta la proprietà dell'oggetto chart OBJPROP_BORDER_COLOR
Font	Ottiene/Imposta la proprietà dell'oggetto chart OBJPROP_FONT
FontSize	Ottiene/Imposta la proprietà dell'oggetto chart OBJPROP_FONTSIZE
ZOrder	Ottiene/Imposta la proprietà dell'oggetto chart OBJPROP_ZORDER
Chart objects event handlers	
OnObjectCreate	CHARTEVENT_OBJECT_CREATE event handler

Elaborazione eventi del Chart	
OnObjectChange	CHARTEVENT_OBJECT_CHANGE event handler
OnObjectDelete	CHARTEVENT_OBJECT_DELETE event handler
OnObjectDrag	CHARTEVENT_OBJECT_DRAG event handler
Event handlers della proprietà change	
OnSetText	"SetText" event handler
OnSetColor	"SetColor" event handler
OnSetColorBackground	"SetColorBackground" event handler
OnSetFont	"SetFont" event handler
OnSetFontSize	"SetFontSize" event handler
OnSetZOrder	"SetZOrder" event handler
Event handler interni	
OnDestroy	Event handler "Distrugge"
OnChange	Event handler "Cambia"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CWnd

Create, Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

Text (Metodo Get)

Ottiene la proprietà (text) OBJPROP_TEXT dell'oggetto chart.

```
string  Text()
```

Valore di ritorno

Il valore della proprietà OBJPROP_TEXT.

Text (Metodo Set)

Imposta la proprietà (text) OBJPROP_TEXT dell'oggetto chart.

```
bool  Text(
   const string  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà OBJPROP_TEXT.

Valore di ritorno

true in caso di successo, altrimenti false.

Color (Metodo Get)

Ottiene la proprietà (colore) OBJPROP_COLOR dell'oggetto chart.

```
color  Color()
```

Valore di ritorno

Il valore della proprietà OBJPROP_COLOR.

Color (Metodo Set)

Imposta la proprietà (colore) OBJPROP_COLOR dell'oggetto chart.

```
bool  Color(
   const color  value      // valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà OBJPROP_COLOR.

Valore di ritorno

true in caso di successo, altrimenti false.

ColorBackground (Metodo Get)

Ottiene la proprietà (colore di sfondo) OBJPROP_BGCOLOR dell'oggetto chart.

```
color  ColorBackground()
```

Valore di ritorno

Il valore della proprietà OBJPROP_BGCOLOR.

ColorBackground (Metoto Set)

Imposta la proprietà (colore di sfondo) OBJPROP_BGCOLOR dell'oggetto chart.

```
bool  ColorBackground(
   const color  value      // valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà OBJPROP_BGCOLOR.

Valore di ritorno

true in caso di successo, altrimenti false.

ColorBorder (Metoto Get)

Ottiene la proprietà (colore bordo) OBJPROP_BORDER_COLOR dell'oggetto chart.

```
color ColorBorder()
```

Valore di ritorno

Il valore della proprietà OBJPROP_BORDER_COLOR.

ColorBorder (Metodo Set)

Imposta la proprietà (colore bordo) OBJPROP_BORDER_COLOR dell'oggetto chart.

```
bool  ColorBorder(
   const color  value      // valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà OBJPROP_BORDER_COLOR.

Valore di ritorno

true in caso di successo, altrimenti false.

Font (Metodo Get)

Ottiene la proprietà (font) OBJPROP_FONT dell'oggetto chart.

```
string  Font()
```

Valore di ritorno

Il valore della proprietà OBJPROP_FONT.

Font (Metoto Set)

Imposta la proprietà (font) OBJPROP_FONT dell'oggetto chart.

```
bool  Font(
   const string  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà OBJPROP_FONT.

Valore di ritorno

true in caso di successo, altrimenti false.

FontSize (Metodo Get)

Ottiene la proprietà (grandezza font) OBJPROP_FONTSIZE dell'oggetto chart.

```
int  FontSize()
```

Valore di ritorno

Il valore della proprietà OBJPROP_FONTSIZE.

FontSize (Metodo Set)

Imposta la proprietà (grandezza font) OBJPROP_FONTSIZE dell'oggetto chart.

```
bool  FontSize(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà OBJPROP_FONTSIZE.

Valore di ritorno

true in caso di successo, altrimenti false.

ZOrder (Metodo Get)

Ottiene la proprietà OBJPROP_ZORDER dell'oggetto chart .

```
long  ZOrder()
```

Valore di ritorno

Il valore della proprietà OBJPROP_FONTSIZE.

ZOrder (Metodo Set)

Imposta la proprietà OBJPROP_ZORDER dell'oggetto chart .

```
bool  ZOrder(
   const long  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà OBJPROP_ZORDER.

Valore di ritorno

true in caso di successo, altrimenti false.

OnObjectCreate

L'event handler CHARTEVENT_OBJECT_CREATE

```
virtual bool  OnObjectCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnObjectChange

L'event handler CHARTEVENT_OBJECT_CHANGE

```
virtual bool  OnObjectChange()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnObjectDelete

L'event handler CHARTEVENT_OBJECT_DELETE

```
virtual bool  OnObjectDelete()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnObjectDrag

L'event handler CHARTEVENT_OBJECT_DRAG

```
virtual bool  OnObjectDrag()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetText

L'event handler del controllo "SetText"(cambio della proprietà OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnSetColor

L'event handler del controllo "SetColor"(cambio della proprietà OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnSetColorBackground

L'event handler del controllo "SetColorBackground"(cambio della proprietà OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnSetFont

L'event handler del controllo "SetFont"(cambio della proprietà OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnSetFontSize

L'event handler del controllo "SetFontSize"(cambio della proprietà OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnSetZOrder

L'event handler del controllo "SetZOrder"(cambio della proprietà OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnDestroy

L'event handler del controllo "Destroy" .

```
virtual bool  OnDestroy()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChange

L'event handler del controllo "Change" .

```
virtual bool  OnChange()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CWndContainer

CWndContainer è una classe base per il controllo complesso (contenente i controlli dipendenti) della libreria Standard.

Descrizione

La Classe CWndContainer implementa metodi di base del controllo complesso.

Dichiarazione

```
class CWndContainer : public CWnd
```

Titolo

```
#include <Controls\WndContainer.mqh>
```

Gerarchia di ereditarietà

CObject

CWnd

CWndContainer

Discendenti diretti

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

Metodi della Classe

Destroy	
Destroy	Distrugge tutti i controlli del contenitore
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
OnMouseEvent	L'event handler CHARTEVENT_MOUSE_MOVE
Accesso al contenitore	
ControlsTotal	Ottiene il numero di controlli nel contenitore
Control	Ottiene il controllo per indice
ControlFind	Ottiene il controllo per ID
Aggiungi/Elimina	
Add	Aggiunge il controllo al contenitore
Delete	Elimina il controllo dal contenitore
Geometry	

Destroy	
Move	Imposta nuove coordinate per tutti i controlli del contenitore
Shift	Esegue lo spostamento relativo delle coordinate per tutti i controlli del contenitore
Identification	
Id	Imposta l'ID per tutti i controlli del contenitore
State	
Enable	Attiva tutti i controlli del contenitore
Disable	Disattiva tutti i controlli del contenitore
Show	Mostra tutti i controlli del contenitore
Hide	Nasconde tutti i controlli del contenitore
Operazioni del mouse	
MouseFocusKill	Elimina il focus del mouse
Operazioni sui file	
Save	Salva le informazioni contenitore in un file
Load	Carica le informazioni del contenitore da file
Event handler interni	
OnResize	Event handler "Ridimensiona"
OnActivate	Event handler "Attiva"
OnDeactivate	Event handler "Disattiva"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Create, Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Destroy

Distrugge tutti i controlli del contenitore.

```
virtual bool  Destroy()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

OnMouseEvent

Mouse event handler.

```
virtual bool  OnMouseEvent(
   const int  x,          // x coordinate
   const int  y,          // y coordinate
   const int  flags       // flags
   )
```

Parametri

x

[in] coordinata X del cursore del mouse rispetto all'angolo in alto a sinistra del grafico.

y

[in] coordinata Y del cursore del mouse rispetto all'angolo in alto a sinistra del grafico.

flags

[in] Flag degli stati dei bottoni del mouse.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

ControlsTotal

Ottiene il numero di controlli nel contenitore.

```
int  ControlsTotal()  const
```

Valore di ritorno

Numero di controlli nel contenitore.

Control

Ottiene il controllo dal contenitore per indice.

```
CWnd*  Control(
   const int  ind      // indice
   )  const
```

Parametri

ind

[in] Indice del controllo richiesto.

Valore di ritorno

Puntatore al controllo, altrimenti NULL se il controllo non è stato trovato.

ControlFind

Ottiene controllo dal contenitore per identificatore.

```
virtual CWnd*  ControlFind(
   const long  id      // id
   )
```

Parametri

id

[in] Control ID.

Valore di ritorno

Puntatore al controllo, altrimenti NULL se il controllo non è stato trovato.

Add

Aggiunge un controllo al contenitore.

```
bool  Add(
   CWnd&  control      // riferimento
   )
```

Parametri

control

[in] Controllo da aggiungere, passato per riferimento.

Valore di ritorno

true in caso di successo, altrimenti false.

Delete

Elimina il controllo dal contenitore.

```
bool  Delete(
   CWnd&  control      // riferimento
   )
```

Parametri

control

[in] Controllo da eliminare, passato per riferimento.

Valore di ritorno

true in caso di successo, altrimenti false.

Move

Imposta nuove coordinate per tutti i controlli del contenitore.

```
virtual bool  Move(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parametri

x

[in] Nuovo coordinata X nell'angolo in alto a sinistra.

y

[in] nuova coordinata Y dell'angolo in alto a sinistra.

Valore di ritorno

true in caso di successo, altrimenti false.

Shift

Esegue lo slittamento relativo delle coordinate per tutti i controlli del contenitore.

```
virtual bool  Shift(
   const int  dx,      // delta x
   const int  dy       // delta y
   )
```

Parametri

dx

[in] Delta X.

dy

[in] Delta Y.

Valore di ritorno

true in caso di successo, altrimenti false.

Id

Imposta l'ID per tutti i controlli del contenitore.

```
virtual long  Id(
   const long  id      // identificatore
   )
```

Parametri

id

[in] Identificatore del gruppo Base.

Valore di ritorno

Numero di identificatori, utilizzato dai controlli del contenitore.

Enable

Attiva tutti i controlli del contenitore.

```
virtual bool  Enable()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Disable

Disattiva tutti i controlli del contenitore.

```
virtual bool  Disable()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Show

Mostra tutti i controlli del contenitore.

```
virtual bool  Show()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Hide

Nasconde tutti i controlli del contenitore.

```
virtual bool  Hide()
```

Valore di ritorno

true in caso di successo, altrimenti false.

MouseFocusKill

Cancella lo stato salvato dei pulsanti del mouse e disattiva tutti i controlli nel contenitore.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parametri

id=CONTROLS_INVALID_ID

[in] Identificatore del controllo, che ha ricevuto il focus del mouse.

Valore di ritorno

Il risultato della disattivazione dei controlli .

Save

Salva le informazioni contenitore in un file.

```
virtual bool  Save(
   const int  file_handle      // handle
   )
```

Parametri

file_handle

[in] Handle del file binario (deve essere aperto per la scrittura).

Valore di ritorno

true in caso di successo, altrimenti false.

Load

Carica le informazioni del contenitore da file

```
virtual bool  Load(
   const int  file_handle      // handle
   )
```

Parametri

file_handle

[in] Handle del file binario (deve essere aperto per la lettura).

Valore di ritorno

true in caso di successo, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnActivate

Il controllo del gestore di eventi (event handler) "Attiva"(Activate).

```
virtual bool  OnActivate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnDeactivate

Il controllo del gestore di eventi (event handler) "Disattiva"(Deactivate).

```
virtual bool  OnDeactivate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

CLabel

CLabel è una classe del controllo semplice, basato sull'oggetto grafico "Etichetta di testo".

Descrizione

CLabel è intesa per la creazione di semplici etichette di testo.

Dichiarazione

```
class CLabel : public CWndObj
```

Titolo

```
#include <Controls\Label.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndObj
CLabel

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo

Create	
Event handlers della proprietà change	
OnSetText	"SetText" event handler
OnSetColor	"SetColor" event handler
OnSetFont	"SetFont" event handler
OnSetFontSize	"SetFontSize" event handler
Event handler interni	
OnCreate	Event handler "Crea"
OnShow	Event handler "Mostra"
OnHide	Event handler "Nascondi"
OnMove	Event handler "Sposta"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Example of creating a panel with text label:

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CLabel"
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
```

```
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CLabel            m_label;                        // CLabel object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- create dependent controls
   bool              CreateLabel(void);
   //--- handlers of the dependent controls events
   void              OnClickLabel(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)

EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
```

```
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateLabel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CLabel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateLabel(void)
  {
//--- coordinates
   int x1=INDENT_RIGHT;
   int y1=INDENT_TOP+CONTROLS_GAP_Y;
   int x2=x1+100;
   int y2=y1+20;
//--- create
   if(!m_label.Create(m_chart_id,m_name+"Label",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_label.Text("Label"))
      return(false);
   if(!Add(m_label))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

```
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CLabel.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnSetText

L'event handler del controllo "SetText" (cambio della proprietà OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColor

L'event handler del controllo "SetColor" (cambio della proprietà OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetFont

L'event handler del controllo "SetFont" (cambio della proprietà OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetFontSize

L'event handler del controllo "SetFontSize" (cambio della proprietà OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnCreate

L'event handler del controllo "Create" .

```
virtual bool  OnCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMove

L'event handler del controllo "Move" .

```
virtual bool  OnMove()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CBmpButton

CBmpButton è una classe del controllo semplice, basata sull'oggetto grafico "Etichetta Bitmap" .

Descrizione

CBmpButton è inteso per la creazione di bottoni con immagine grafica.

Dichiarazione

```
class CBmpButton : public CWndObj
```

Titolo

```
#include <Controls\BmpButton.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndObj
CBmpButton

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo

Create	
Proprietà	
Border	Ottiene/Imposta la proprietà "Bordo" del controllo
BmpNames	Imposta il nome del file BMP del controllo
BmpOffName	Ottiene/Imposta il nome del file BMP per lo stato OFF
BmpOnName	Ottiene/Imposta il nome del file BMP per lo stato ON
BmpPassiveName	Ottiene/Imposta il nome del file BMP per lo stato passivo
BmpActiveName	Ottiene/Imposta il nome del file BMP per lo stato attivo
State	
Pressed	Ottiene/Imposta lo stato del controllo
Locking	Ottiene/Imposta la proprietà "Blocco" del controllo
Event handler interni	
OnSetZOrder	"SetZOrder" event handler
OnCreate	Event handler "Crea"
OnShow	Event handler "Mostra"
OnHide	Event handler "Nascondi"
OnMove	Event handler "Sposta"
OnChange	Event handler "Cambia"
OnActivate	Event handler "Attiva"
OnDeactivate	Event handler "Disattiva"
OnMouseDown	Event handler "Mouse giù"
OnMouseUp	Event handler "Mouse sù"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Example of creating a panel with Bitmap label:

```
//+------------------------------------------------------------------+
//|                                              ControlsBmpButton.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CBmpButton"
#include <Controls\Dialog.mqh>
#include <Controls\BmpButton.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CBmpButton        m_bmpbutton1;                  // CBmpButton object
   CBmpButton        m_bmpbutton2;                  // CBmpButton object
```

```
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateBmpButton1(void);
   bool              CreateBmpButton2(void);
   //--- handlers of the dependent controls events
   void              OnClickBmpButton1(void);
   void              OnClickBmpButton2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_bmpbutton1,OnClickBmpButton1)
ON_EVENT(ON_CLICK,m_bmpbutton2,OnClickBmpButton2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateBmpButton1())
      return(false);
   if(!CreateBmpButton2())
      return(false);
//--- succeed
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton1" button                                   |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton1.Create(m_chart_id,m_name+"BmpButton1",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton1.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton2" fixed button                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton2.Create(m_chart_id,m_name+"BmpButton2",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton2.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton2))
      return(false);
   m_bmpbutton2.Locking(true);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton1(void)
  {
   Comment(__FUNCTION__);
  }
```

```
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton2(void)
  {
   if(m_bmpbutton2.Pressed())
      Comment(__FUNCTION__+" State of the control is: On");
   else
      Comment(__FUNCTION__+" State of the control is: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CBmpButton.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

Border (Metodo Get)

Ottiene la proprietà del controllo "Bordo" (larghezza del bordo) .

```
int  Border()  const
```

Valore di ritorno

La "proprietà Bordo(Border)".

Border (Metodo Set)

Imposta la proprietà del controllo "Bordo" (larghezza del bordo) .

```
bool  Border(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della "proprietà Bordo (Border)".

Valore di ritorno

true in caso di successo, altrimenti false.

BmpNames

Imposta il nome del file BMP del controllo

```
bool  BmpNames(
   const string  off="",      // nome del file
   const string  on=""        // nome del file
   )
```

Parametri

off=""

[in] Nome del file bmp per lo stato OFF.

on=""

[in] Nome del file bmp per lo stato ON.

Valore di ritorno

true in caso di successo, altrimenti false.

BmpOffName (Metodo Get)

Ottiene il nome del file BMP per lo stato OFF.

```
string  BmpOffName()  const
```

Valore di ritorno

Nome del file BMP per lo stato OFF.

BmpOffName (Metodo Set)

Imposta il nome del file BMP per lo stato OFF.

```
bool  BmpOffName(
   const string  name      // nome del file
   )
```

Parametri

name

[in] Nome del file bmp per lo stato OFF.

Valore di ritorno

true in caso di successo, altrimenti false.

BmpOnName (Metodo Get)

Ottiene il nome del file BMP per lo stato ON.

```
string  BmpOnName()  const
```

Valore di ritorno

Nome del file BMP per lo stato ON.

BmpOnName (Metodo Set)

Imposta il nome del file BMP per lo stato ON.

```
bool  BmpOnName(
   const string  name      // nome del file
   )
```

Parametri

name

[in] Nome del file bmp per lo stato ON.

Valore di ritorno

true in caso di successo, altrimenti false.

BmpPassiveName (Metodo Get)

Ottiene il nome del file BMP per lo stato di controllo passivo.

```
string  BmpPassiveName()  const
```

Valore di ritorno

Nome di file BMP per lo stato passivo di controllo.

BmpPassiveName (Metodo Set)

Imposta il nome del file BMP per lo stato passivo.

```
bool  BmpPassiveName(
   const string  name      // nome del file
   )
```

Parametri

name

[in] Nome del file BMP per lo stato passivo di controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

BmpActiveName (Metodo Get)

Ottiene il nome del file BMP per lo stato attivo.

```
string  BmpActiveName()  const
```

Valore di ritorno

Nome del file BMP per lo stato attivo.

Nota

Il controllo si attiva quando il cursore del mouse passa su di esso.

BmpActiveName (Metodo Set)

Imposta il nome del file BMP per lo stato attivo.

```
bool  BmpActiveName(
   const string  name      // nome del file
   )
```

Parametri

name

[in] Nome del file BMP per lo stato attivo.

Valore di ritorno

true in caso di successo, altrimenti false.

Pressed (Metodo Get)

Ottiene lo stato ("Pressed"(premuto) proprietà) del controllo.

```
bool  Pressed()  const
```

Valore di ritorno

Stato del controllo.

Pressed (Metodo Set)

Imposta lo stato ("Pressed"(premuto) proprietà) del controllo.

```
bool  Pressed(
   const bool  pressed      // nuovo stato
   )
```

Parametri

pressed

[in] Nuovo stato del controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

Locking (Metodo Get)

Ottiene la proprietà "Blocco" del controllo

```
bool  Locking()  const
```

Valore di ritorno

Il valore della proprietà "Locking"(Blocco).

Locking (Metodo Set)

Imposta il valore della proprietà "Blocco" del controllo.

```
void  Locking(
   const bool  locking      // nuovo valore
   )
```

Parametri

locking

[in] Nuovo valore della proprietà "Locking".

Valore di ritorno

Nessuno.

OnSetZOrder

L'event handler del controllo "SetZOrder" (cambio della proprietà OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnCreate

L'event handler del controllo "Create" .

```
virtual bool  OnCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMove

L'event handler del controllo "Move" .

```
virtual bool  OnMove()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChange

L'event handler del controllo "Change" .

```
virtual bool  OnChange()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnActivate

Il controllo del gestore di eventi (event handler) "Attiva"(Activate).

```
virtual bool  OnActivate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnDeactivate

Il controllo del gestore di eventi (event handler) "Disattiva"(Deactivate).

```
virtual bool  OnDeactivate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMouseDown

L'event handler del controllo "MouseDown" .

```
virtual bool  OnMouseDown()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "MouseDown" si verifica quando si preme il pulsante sinistro del mouse sul controllo.

OnMouseUp

Il gestore dell'evento (event handler) "MouseUp" (rilascio del pulsante sinistro del mouse) .

```
virtual bool  OnMouseUp()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "MouseDown" si verifica quando si rilascia il pulsante sinistro del mouse sul controllo.

CButton

CButton è una classe del semplice controllo, basata sull'oggetto grafico "Button".

Descrizione

La Classe CButton è intesa per la creazione di semplici bottoni.

Dichiarazione

```
class CButton : public CWndObj
```

Titolo

```
#include <Controls\Button.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndObj
CButton

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo

Create	
State	
Pressed	Ottiene/Imposta la proprietà "Pressed"(premuto)
Locking	Ottiene/Imposta la proprietà "Locking"(blocco)
Event handlers della proprietà change	
OnSetText	"SetText" event handler
OnSetColor	"SetColor" event handler
OnSetColorBackground	"SetColorBackground" event handler
OnSetColorBorder	"SetColorBorder" event handler
OnSetFont	"SetFont" event handler
OnSetFontSize	"SetFontSize" event handler
Event handler interni	
OnCreate	Event handler "Crea"
OnShow	Event handler "Mostra"
OnHide	Event handler "Nascondi"
OnMove	Event handler "Sposta"
OnResize	Event handler "Ridimensiona"
OnMouseDown	Event handler "Mouse giù"
OnOnMouseUp	Event handler "Mouse sù"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Example of creating a panel with button:

```
//+------------------------------------------------------------------+
```

```
//|                                                ControlsButton.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CButton"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CButton           m_button1;                     // the button object
   CButton           m_button2;                     // the button object
   CButton           m_button3;                     // the fixed button object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
```

```
protected:
   //--- create dependent controls
   bool              CreateButton1(void);
   bool              CreateButton2(void);
   bool              CreateButton3(void);
   //--- handlers of the dependent controls events
   void              OnClickButton1(void);
   void              OnClickButton2(void);
   void              OnClickButton3(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_button1,OnClickButton1)
ON_EVENT(ON_CLICK,m_button2,OnClickButton2)
ON_EVENT(ON_CLICK,m_button3,OnClickButton3)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateButton1())
      return(false);
   if(!CreateButton2())
      return(false);
   if(!CreateButton3())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button1" button                                      |
```

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button1.Create(m_chart_id,m_name+"Button1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button1.Text("Button1"))
      return(false);
   if(!Add(m_button1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button2" button                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button2.Create(m_chart_id,m_name+"Button2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button2.Text("Button2"))
      return(false);
   if(!Add(m_button2))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button3" fixed button                                |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton3(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
```

```
   if(!m_button3.Create(m_chart_id,m_name+"Button3",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button3.Text("Locked"))
      return(false);
   if(!Add(m_button3))
      return(false);
   m_button3.Locking(true);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton2(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton3(void)
  {
   if(m_button3.Pressed())
      Comment(__FUNCTION__+" State of the control: On");
   else
      Comment(__FUNCTION__+" State of the control: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
```

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CButton.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

Pressed (Metodo Get)

Ottiene lo stato ("Pressed"(premuto) proprietà) del controllo.

```
bool  Pressed()  const
```

Valore di ritorno

Stato del controllo.

Pressed (Metodo Set)

Imposta lo stato ("Pressed"(premuto) proprietà) del controllo.

```
bool  Pressed(
   const bool  pressed      // nuovo stato
   )
```

Parametri

pressed

[in] Nuovo stato del controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

Locking (Metodo Get)

Ottiene la proprietà "Blocco" del controllo

```
bool  Locking()  const
```

Valore di ritorno

Il valore della proprietà "Locking"(Blocco).

Locking (Metodo Set)

Imposta nuovo valore della proprietà "Locking" del controllo.

```
void  Locking(
   const bool  locking      // nuovo valore
   )
```

Parametri

locking

[in] Nuovo valore della proprietà "Locking".

Valore di ritorno

Nessuno.

OnSetText

L'event handler del controllo "SetText" (cambio della proprietà OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColor

L'event handler del controllo "SetColor" (cambio della proprietà OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColorBackground

L'event handler del controllo "SetColorBackground" (cambio della proprietà OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColorBorder

L'event handler del controllo "SetColorBorder" (cambio della proprietà OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetFont

L'event handler del controllo "SetFont" (cambio della proprietà OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetFontSize

L'event handler del controllo "SetFontSize" (cambio della proprietà OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnCreate

L'event handler del controllo "Create" .

```
virtual bool  OnCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMove

L'event handler del controllo "Move" .

```
virtual bool  OnMove()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMouseDown

L'event handler del controllo "MouseDown" .

```
virtual bool  OnMouseDown()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "MouseDown" si verifica quando si preme il pulsante sinistro del mouse sul controllo.

OnMouseUp

Il gestore dell'evento (event handler) "MouseUp" (rilascio del pulsante sinistro del mouse) .

```
virtual bool  OnMouseUp()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "MouseDown" si verifica quando si rilascia il pulsante sinistro del mouse sul controllo.

CEdit

CEdit è una classe del semplice controllo, basato sulla "Modifica"(Edit) dell'oggetto chart.

Descrizione

La classe CEdit è intesa per la creazione di controlli, in cui l'utente può immettere il testo.

Dichiarazione

```
class CEdit : public CWndObj
```

Titolo

```
#include <Controls\Edit.mqh>
```

Gerarchia di ereditarietà

CObject
  CWnd
    CWndObj
      CEdit

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo

Create	
Proprietà	
ReadOnly	Ottiene/Imposta la proprietà "ReadOnly"(Sola Lettura)
TextAlign	Ottiene/Imposta la proprietà "TextAlign"
Event handlers dell'oggetto Chart	
OnObjectEndEdit	L'event handler (virtual) CHARTEVENT_OBJECT_ENDEDIT
Event handlers della proprietà change	
OnSetText	"SetText" event handler
OnSetColor	"SetColor" event handler
OnSetColorBackground	"SetColorBackground" event handler
OnSetColorBorder	"SetColorBorder" event handler
OnSetFont	"SetFont" event handler
OnSetFontSize	"SetFontSize" event handler
OnSetZOrder	"SetZOrder" event handler
Event handler interni	
OnCreate	Event handler "Crea"
OnShow	Event handler "Mostra"
OnHide	Event handler "Nascondi"
OnMove	Event handler "Sposta"
OnResize	Event handler "Ridimensiona"
OnChange	Event handler "Cambia"
OnClick	Event handler "Click"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndObj

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Example of creating a panel with Edit control:

```
//+------------------------------------------------------------------+
//|                                                ControlsEdit.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CEdit"
#include <Controls\Dialog.mqh>
#include <Controls\Edit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CEdit             m_edit;                          // CEdit object

public:
```

```
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler

protected:
   //--- create dependent controls
   bool              CreateEdit(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the display field                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateEdit(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=ClientAreaWidth()-INDENT_RIGHT;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_edit.Create(m_chart_id,m_name+"Edit",m_subwin,x1,y1,x2,y2))
      return(false);
//--- allow editing the content
```

```
   if(!m_edit.ReadOnly(false))
      return(false);
   if(!Add(m_edit))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CEdit.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

ReadOnly (Metodo Get)

Ottiene la proprietà del controllo "ReadOnly".

```
bool  ReadOnly()
```

Valore di ritorno

Il valore della proprietà "ReadOnly" .

ReadOnly (Metodo Set)

Imposta il valore della proprietà "ReadOnly" del controllo.

```
bool  ReadOnly(
   const bool  flag      // nuovi valori
   )
```

Parametri

flag

[in] Nuovo valore della proprietà"ReadOnly".

Valore di ritorno

true in caso di successo, altrimenti false.

TextAlign (Metodo Get)

Ottiene il valore della proprietà "TextAlign" (modalità allineamento del testo) del controllo.

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valore di ritorno

Valore della proprietà "TextAlign" del controllo.

TextAlign (Set method)

Imposta nuovo valore della proprietà "TextAlign" (modalità di allineamento del testo) del controllo.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // nuovo valore
   )
```

Parametri

align

[in] Nuovo valore della proprietà "TextAlign".

Valore di ritorno

true in caso di successo, false se il templete non è stato cambiato.

OnObjectEndEdit

L'event handler (virtual) CHARTEVENT_OBJECT_ENDEDIT

```
virtual bool  OnObjectEndEdit()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetText

L'event handler del controllo "SetText" (cambio della proprietà OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColor

L'event handler del controllo "SetColor" (cambio della proprietà OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColorBackground

L'event handler del controllo "SetColorBackground" (cambio della proprietà OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColorBorder

Il controllo "SetColorBorder" (event handler del cambiamento della proprietà OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetFont

L'event handler del controllo "SetFont" (cambio della proprietà OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetFontSize

L'event handler del controllo "SetFontSize" (cambio della proprietà OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetZOrder

L'event handler del controllo "SetZOrder" (cambio della proprietà OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnCreate

L'event handler del controllo "Create" .

```
virtual bool  OnCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMove

L'event handler del controllo "Move" .

```
virtual bool  OnMove()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChange

L'event handler del controllo "Change" .

```
virtual bool  OnChange()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnClick

L'event handler del controllo "Click" (clic del mouse sul tasto sinistro).

```
virtual bool  OnClick()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CPanel

CPanel è una classe del controllo semplice, sulla base dell'oggetto grafico "Etichetta rettangolo".

Descrizione

LA classe CPanel è intesa a combinare i controlli con funzioni simili nel gruppo.

Dichiarazione

```
class CPanel : public CWndObj
```

Titolo

```
#include <Controls\Panel.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndObj
CPanel

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo

Create	
Proprietà dell'oggetto Chart	
BorderType	Ottiene la proprietà "BorderType"(Tipo di Bordo) dell'oggetto chart
Event handlers dell'oggetto Chart	
OnSetText	"SetText" event handler
OnSetColorBackground	"SetColorBackground" event handler
OnSetColorBorder	"SetColorBorder" event handler
Event handler interni	
OnCreate	Event handler "Crea"
OnShow	Event handler "Mostra"
OnHide	Event handler "Nascondi"
OnMove	Event handler "Sposta"
OnResize	Event handler "Ridimensiona"
OnChange	Event handler "Cambia"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Example of creating a panel with Rectangle label:

```
//+------------------------------------------------------------------+
//|                                                ControlsPanel.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CPanel"
#include <Controls\Dialog.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const

protected:
   //--- create dependent controls
   bool              CreatePanel(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
```

```
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePanel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CPanel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePanel(void)
  {
//--- coordinates
   int x1=20;
   int y1=20;
   int x2=ExtDialog.Width()/3;
   int y2=ExtDialog.Height()/3;
//--- create
   if(!my_white_border.Create(0,ExtDialog.Name()+"MyWhiteBorder",m_subwin,x1,y1,x2,y2)
      return(false);
   if(!my_white_border.ColorBackground(CONTROLS_DIALOG_COLOR_BG))
      return(false);
   if(!my_white_border.ColorBorder(CONTROLS_DIALOG_COLOR_BORDER_LIGHT))
      return(false);
   if(!ExtDialog.Add(my_white_border))
      return(false);
   my_white_border.Alignment(WND_ALIGN_CLIENT,0,0,0,0);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//---
CPanel   my_white_border;        // object CPanel
bool     pause=true;             // true - pause
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

```
  {
//---
   EventSetTimer(3);
   pause=true;
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//|  Timer                                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
   pause=!pause;
  }
```

Create

Crea nuovo controllo CPanel.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

BorderType (Metodo Get)

Ottiene la proprietà "BorderType" dell'oggetto chart.

```
ENUM_BORDER_TYPE  BorderType()
```

Valore di ritorno

Il valore della proprietà "BorderType".

BorderType (Metodo Set)

Imposta il nuovo valore della proprietà "BorderType" dell'oggetto chart.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // valore
   )
```

Parametri

type

[in] Nuovo valore della proprietà "BorderType".

Valore di ritorno

true in caso di successo, altrimenti false.

OnSetText

L'event handler del controllo "SetText" (cambio della proprietà OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColorBackground

L'event handler del controllo "SetColorBackground" (cambio della proprietà OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnSetColorBorder

Il controllo "SetColorBorder" (event handler del cambiamento della proprietà OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnCreate

L'event handler del controllo "Create" .

```
virtual bool  OnCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMove

L'event handler del controllo "Move" .

```
virtual bool  OnMove()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChange

L'event handler del controllo "Change" .

```
virtual bool  OnChange()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CPicture

CPicture è una classe del controllo semplice, basata sull'oggetto chart "Bitmap Label".

Descrizione

La Classe cPicture è intesa per la creazione di immagini grafiche semplici.

Dichiarazione

```
class CPicture : public CWndObj
```

Titolo

```
#include <Controls\Picture.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndObj
CPicture

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo

Create	
Proprietà dell'oggetto Chart	
Border	Ottiene/Imposta la larghezza del bordo dell'oggetto chart
BmpName	Ottiene/Imposta il nome del file bmp del controllo
Eventi interni	
OnCreate	Event handler "Crea"
OnShow	Event handler "Mostra"
OnHide	Event handler "Nascondi"
OnMove	Event handler "Sposta"
OnChange	Event handler "Cambia"

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Metodi ereditati dalla classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Example of creating a panel with Bitmap label:

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CPicture"
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
```

```
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CPicture          m_picture;                      // CPicture object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreatePicture(void);
   //--- handlers of the dependent controls events
   void              OnClickPicture(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_picture,OnClickPicture)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
```

```
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePicture())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Picture"                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePicture(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+32;
   int y2=y1+32;
//--- create
   if(!m_picture.Create(m_chart_id,m_name+"Picture",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set the name of bmp files to display the CPicture control
   m_picture.BmpName("\\Images\\euro.bmp");

   if(!Add(m_picture))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
  {
   Comment(__FUNCTION__);
  }
```

```
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CPicture.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

Border (Metodo Get)

Ottiene la proprietà del controllo "Bordo" (larghezza del bordo) .

```
int  Border()  const
```

Valore di ritorno

La "proprietà Bordo(Border)".

Border (Metodo Set)

Imposta la proprietà del controllo "Bordo" (larghezza del bordo) .

```
bool  Border(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della "proprietà Bordo (Border)".

Valore di ritorno

true in caso di successo, altrimenti false.

BmpName (Metodo Get)

Ottiene il nome del file BMP del controllo.

```
string  BmpName()  const
```

Valore di ritorno

Nome del file bmp del controllo.

BmpName (Metodo Set)

Imposta il nome del file bmp del controllo.

```
bool  BmpName(
   const string  name      // nome del file
   )
```

Parametri

name

[in] Nome del file bmp del controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

OnCreate

L'event handler del controllo "Create" .

```
virtual bool  OnCreate()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnMove

L'event handler del controllo "Move" .

```
virtual bool  OnMove()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChange

L'event handler del controllo "Change" .

```
virtual bool  OnChange()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CScroll

CScroll è una classe base per la creazione di barre di scorrimento.

Descrizione

CScroll è un controllo complesso (con controlli dipendenti), che contiene la funzionalità di base per la creazione di barre di scorrimento. La classe di base stessa non è utilizzata come controllo separato, due dei suoi eredi (classi CScrollV e CScrollH) vengono utilizzati come controlli.

Dichiarazione

```
class CScroll : public CWndContainer
```

Titolo

```
#include <Controls\Scrolls.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CScroll

Discendenti diretti

CScrollH, CScrollV

Metodi della Classe

Create	
Create	Crea il controllo
Event handlers dell'oggetto Chart	
OnEvent	Event Handler di tutti gli eventi del chart
Proprietà	
MinPos	Ottiene/Imposta la posizione minima
MaxPos	Ottiene/Imposta la posizione massima
CurrPos	Ottiene/Imposta la posizione corrente
Creazione controlli dipendenti	
CreateBack	Crea il bottone sfondo
CreateInc	Crea bottone di incremento della barra di scorrimento

Create	
CreateDec	Crea bottone di decremento della barra di scorrimento
CreateThumb	Crea il bottone del pollice (può essere trascinato) della barra di scorrimento
Event handlers dei controlli Dependent	
OnClickInc	Event handler, utilizzato per la manipolazione degli eventi dei bottoni di incremento
OnClickDec	Event handler, utilizzato per la manipolazione degli eventi dei bottoni di decremento
Event handler interni	
OnShow	Event handler "Crea"
OnHide	Event handler "Nascondi"
OnChangePos	"ChangePosition" event handler
Object drag handlers	
OnThumbDragStart	"ThumbDragStart" event handler
OnThumbDragProcess	"ThumbDragProcess" event handler
OnThumbDragEnd	"ThumbDragEnd" event handler
Position	
CalcPos	Ottiene scorrimento posizione della barra per coordinare

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Create

Crea nuovo controllo CScroll.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

MinPos (Metodo Get)

Ottiene il valore di "MinPos" (posizione minima) del controllo CScroll.

```
int  MinPos()  const
```

Valore di ritorno

Nuovo valore della proprietà "MinPos".

MinPos (Metodo Set)

Imposta il valore di "MinPos" (posizione minima) del controllo CScroll.

```
void  MinPos(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà "MinPos".

Valore di ritorno

Nessuno.

MaxPos (Metodo Get)

Ottiene il valore di "MaxPos" (posizione massima) del controllo CScroll.

```
int  MaxPos()  const
```

Valore di ritorno

Nuovo valore della proprietà "MaxPos".

MaxPos (Metodo Set)

Imposta il valore di "MaxPos" (posizione massima) del controllo CScroll.

```
void  MaxPos(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà "MaxPos".

Valore di ritorno

Nessuno.

CurrPos (Metodo Get)

Ottiene il valore di "CurrPos" (posizione attuale) del controllo CScroll.

```
int  CurrPos()  const
```

Valore di ritorno

Nuovo valore della proprietà "CurrPos".

CurrPos (Metodo Set)

Imposta il valore di "CurrPos" (posizione attuale) del controllo CScroll.

```
void  CurrPos(
   const int  value      // nuovo valore
   )
```

Parametri

value

[In] Nuovo valore della proprietà "CurrPos".

Valore di ritorno

Nessuno.

CreateBack

Crea il bottone sfondo del controllo CScroll.

```
virtual bool  CreateBack()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateInc

Crea bottone di incremento del controllo CScroll.

```
virtual bool  CreateInc()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateDec

Crea bottone di decremento del controllo CScroll.

```
virtual bool  CreateDec()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateThumb

Crea il bottone del pollice (può essere trascinato) del controllo CScroll.

```
virtual bool  CreateThumb()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnClickInc

L'event handler del controllo "ClickInc" (click tasto sinistro del mouse sul bottone incremento).

```
virtual bool  OnClickInc()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnClickDec

L'event handler del controllo "ClickDec" (click tasto sinistro del mouse sul bottone decremento).

```
virtual bool  OnClickDec()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnShow

L'event handler del controllo "Show" .

```
ùvirtual bool  OnShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnHide

L'event handler del controllo "Hide" .

```
virtual bool  OnHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChangePos

L'event handler del controllo "ChangePos" (cambio posizione).

```
virtual bool  OnChangePos()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnThumbDragStart

L'event handler del controllo "ThumbDragStart" (processo di trascinamento avviato).

```
virtual bool  OnThumbDragStart()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "ThumbDragStart" si verifica all'inizio della operazione di trascinamento.

OnThumbDragProcess

L'event handler del controllo "ThumbDragProcess" .

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parametri

x

[in] Attuale coordinata X del cursore del mouse.

y

[in] Attuale coordinata Y del cursore del mouse.

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il "ThumbDragProcess" si verifica quando il controllo barra di scorrimento (bottone pollice) viene spostato.

OnThumbDragEnd

L'event handler del controllo "ThumbDragEnd" (processo di trascinamento concluso).

```
virtual bool  OnThumbDragEnd()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il "ThumbDragEnd" si verifica quando l'operazione di trascinamento del controllo barra di scorrimento (bottone pollice) è finito.

CalcPos

Ottiene scorrimento posizione della barra per coordinare.

```
virtual int  CalcPos(
   const int      coord      // coordinate
   )
```

Parametri

coord

[in] Coordinate della scroll bar.

Valore di ritorno

Oosizione della barra di scorrimento.

CScrollV

CScrollV è una classe del controllo complesso "Barra di scorrimento verticale".

Descrizione

La Classe CScrollV è intesa per la creazione di barre di scorrimento verticali.

Dichiarazione

```
class CScrollV : public CScroll
```

Titolo

```
#include <Controls\Scrolls.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CScroll
CScrollV

Result of the code provided below:

Metodi della Classe

Controlli dipendenti	
CreateInc	Crea il bottone di incremento della barra di scorrimento
CreateDec	Crea il bottone di decremento della barra di scorrimento
CreateThumb	Crea bottone pollice della barra di scorrimento (può essere trascinato)
Event handler interni	
OnResize	Event handler "Ridimensiona"
OnChangePos	"ChangePosition" event handler
Event handlers trascinamento	
OnThumbDragStart	"ThumbDragStart" event handler
OnThumbDragProcess	"ThumbDragProcess" event handler
OnThumbDragEnd	"ThumbDragEnd" event handler
Position	
CalcPos	Ottiene scorrimento posizione della barra per coordinare

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Metodi ereditati dalla classe CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Example of creating a panel with vertical scrollbar:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollV.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CScrollV"
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollV          m_scroll_v;                     // CScrollV object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateScrollV(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
```

```
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollV())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsV object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollV(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+18;
   int y2=y1+LIST_HEIGHT;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollV",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set up the scrollbar
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
```

```
      return(false);
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
```

```
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

CreateInc

Crea pulsante di incremento del controllo.

```
virtual bool  CreateInc()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateDec

Crea pulsante di decremento del controllo.

```
virtual bool  CreateDec()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateThumb

Crea il bottone del pollice (può essere trascinato) del controllo.

```
virtual bool  CreateThumb()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChangePos

L'event handler del controllo "ChangePos" (cambio posizione).

```
virtual bool  OnChangePos()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnThumbDragStart

L'event handler del controllo "ThumbDragStart" (processo di trascinamento avviato).

```
virtual bool  OnThumbDragStart()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "ThumbDragStart" si verifica all'inizio della operazione di trascinamento.

OnThumbDragProcess

L'event handler del controllo "ThumbDragProcess" .

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parametri

x

[in] Attuale coordinata X del cursore del mouse.

y

[in] Attuale coordinata Y del cursore del mouse.

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il "ThumbDragProcess" si verifica quando il controllo barra di scorrimento (bottone pollice) viene spostato.

OnThumbDragEnd

L'event handler del controllo "ThumbDragEnd" (processo di trascinamento concluso).

```
virtual bool  OnThumbDragEnd()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il "ThumbDragEnd" si verifica quando l'operazione di trascinamento del controllo barra di scorrimento (bottone pollice) è finito.

CalcPos

Ottiene scorrimento posizione della barra per coordinare.

```
virtual int  CalcPos(
   const int      coord      // coordinate
   )
```

Parametri

coord

[in] Coordinate della scroll bar.

Valore di ritorno

Oosizione della barra di scorrimento.

CScrollH

CScrollH è una classe del controllo complesso "Barra di scorrimento orizzontale".

Descrizione

CScrollH è inteso per la creazione di barre di scorrimento orizzontali.

Dichiarazione

```
class CScrollH : public CScroll
```

Titolo

```
#include <Controls\Scrolls.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CScroll
CScrollH

Result of the code provided below:

Metodi della Classe

Controlli dipendenti	
CreateInc	Crea il bottone di incremento della barra di scorrimento
CreateDec	Crea il bottone di decremento della barra di scorrimento
CreateThumb	Crea bottone pollice della barra di scorrimento (può essere trascinato)
Event handler interni	
OnResize	Event handler "Ridimensiona"
OnChangePos	"ChangePosition" event handler
Event handlers trascinamento	
OnThumbDragStart	"ThumbDragStart" event handler
OnThumbDragProcess	"ThumbDragProcess" event handler
OnThumbDragEnd	"ThumbDragEnd" event handler
Position	
CalcPos	Ottiene scorrimento posizione della barra per coordinare

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Metodi ereditati dalla classe CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Example of creating a panel with horizontal scrollbar:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollH.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CScrollH"
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollH          m_scroll_v;                     // CScrollH object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateScrollsH(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
```

```
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollsH())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsH object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollsH(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+3*BUTTON_WIDTH;
   int y2=y1+18;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollsH",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set up the scrollbar
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
```

```
      return(false);
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
```

```
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

CreateInc

Crea pulsante di incremento del controllo.

```
virtual bool  CreateInc()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateDec

Crea pulsante di decremento del controllo.

```
virtual bool  CreateDec()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateThumb

Crea il bottone del pollice (può essere trascinato) del controllo.

```
virtual bool  CreateThumb()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChangePos

L'event handler del controllo "ChangePos" (cambio posizione).

```
virtual bool  OnChangePos()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnThumbDragStart

L'event handler del controllo "ThumbDragStart" (processo di trascinamento avviato).

```
virtual bool  OnThumbDragStart()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "ThumbDragStart" si verifica all'inizio della operazione di trascinamento.

OnThumbDragProcess

L'event handler del controllo "ThumbDragProcess" .

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parametri

x

[in] Attuale coordinata X del cursore del mouse.

y

[in] Attuale coordinata Y del cursore del mouse.

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il "ThumbDragProcess" si verifica quando il controllo barra di scorrimento (bottone pollice) viene spostato.

OnThumbDragEnd

L'event handler del controllo "ThumbDragEnd" (processo di trascinamento concluso).

```
virtual bool  OnThumbDragEnd()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il "ThumbDragEnd" si verifica quando l'operazione di trascinamento del controllo barra di scorrimento (bottone pollice) è finito.

CalcPos

Ottiene scorrimento posizione della barra per coordinare.

```
virtual int  CalcPos(
   const int      coord      // coordinate
   )
```

Parametri

coord

[in] Coordinate della scroll bar.

Valore di ritorno

Oosizione della barra di scorrimento.

CWndClient

CWndClient è una classe del controllo complesso "Area Client" (con i controlli dipendenti). Si tratta di una classe di base per la creazione di un'area a barre di scorrimento.

Descrizione

La classe CWndClient implementa la funzionalità per la creazione di un'area client con barre di scorrimento.

Dichiarazione

```
class CWndClient : public CWndContainer
```

Titolo

```
#include <Controls\WndClient.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CWndClient

Discendenti diretti

CCheckGroup, CListView, CRadioGroup

Metodi della Classe

Create	
Create	Crea il controllo
Chart event handler	
OnEvent	Event Handler di tutti gli eventi del chart
Proprietà	
ColorBackground	Imposta colore di sfondo
ColorBorder	Imposta il colore del bordo
BorderType	Imposta tipo di bordo
Impostazioni	
VScrolled	Ottiene/Imposta la flag che indica che la barra di scorrimento verticale viene utilizzata
HScrolled	Ottiene/Imposta la flag che indica che la barra di scorrimento orizzontale viene utilizzata

Create	
Controlli dipendenti	
CreateBack	Crea sfondo per la barra di scorrimento
CreateScrollV	Crea barra di scorrimento verticale
CreateScrollH	Crea barra di scorrimento orizzontale
Event handler interni	
OnResize	Event handler "Ridimensiona"
Event handlers dei controlli Dependent	
OnVScrollShow	Event handler "Show" (virtual) del controllo dipendente VScroll
OnVScrollHide	Event handler "Hide" (virtual) del controllo dipendente VScroll
OnHScrollShow	Event handler (virtuale) "Show" del controllo dipendente HScroll
OnHScrollHide	Event handler "Hide" (virtual) del controllo dipendente HScroll
OnScrollLineDown	Event handler "ScrollLineDown" (virtual) del controllo dipendente VScroll
OnScrollLineUp	Event handler "ScrollLineUp" (virtual) del controllo dipendente VScroll
OnScrollLineLeft	Event handler "ScrollLineLeft" (virtual) del controllo dipendente HScroll
OnScrollLineRight	Event handler "ScrollLineRight" (virtual) del controllo dipendente HScroll
Ridimensiona	
Rebound	Imposta le nuove coordinate del controllo utilizzando le coordinate della classe CRect

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Create

Crea nuovo controllo CWndClient.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

ColorBackground

Imposta il colore di sfondo del controllo.

```
bool  ColorBackground(
   const color  value      // nuovo colore
   )
```

Parametri

value

[in] Nuovo colore di sfondo del controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

ColorBorder

Imposta il colore del bordo del controllo.

```
bool  ColorBorder(
   const color  value      // colore
   )
```

Parametri

value

[in] Nuovo colore del bordo del controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

BorderType

Imposta tipo di bordo del controllo.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // tipo di bordo
   )
```

Parametri

type

[in] Tipo di bordo del controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

VScrolled (Metodo Get)

Ottiene la flag, che indica che la barra di scorrimento verticale viene utilizzata.

```
bool  VScrolled()
```

Valore di ritorno

true, se si utilizza la barra di scorrimento verticale, altrimenti false.

VScrolled (Metodo Set)

Imposta la flag, che indica che la barra di scorrimento verticale viene utilizzata.

```
bool VScrolled(
   const bool  flag      // flag
   )
```

Parametri

flag

[in] Flag.

Valore di ritorno

true in caso di successo, altrimenti false.

HScrolled (Metodo Get)

Ottiene la flag, che indica che la barra di scorrimento orizzontale viene utilizzata.

```
bool  HScrolled()
```

Valore di ritorno

true, se si utilizza la barra di scorrimento orizzontale, altrimenti false.

HScrolled (Metodo Set)

Imposta la bandiera, che indica che la barra di scorrimento orizzontale viene utilizzata.

```
bool  HScrolled(
   const bool  flag      // flag
   )
```

Parametri

flag

[in] Flag.

Valore di ritorno

true in caso di successo, altrimenti false.

CreateBack

Crea bottone di sfondo del controllo.

```
virtual bool  CreateBack()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateScrollV

Crea barra di scorrimento verticale.

```
virtual bool  CreateScrollV()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateScrollH

Crea barra di scorrimento orizzontale.

```
virtual bool  CreateScrollH()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnVScrollShow

Il gestore dell'evento (event handler) "VScrollShow" (mostrare la barra di scorrimento) .

```
virtual bool  OnVScrollShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnVScrollHide

Il gestore dell'evento (event handler) "VScrollHide" (nascondere la barra di scorrimento) .

```
virtual bool  OnVScrollHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnHScrollShow

L'event handler del controllo "HScrollShow" (mostra barra di scorrimento orizzontale) .

```
virtual bool  OnHScrollShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnHScrollHide

L'event handler del controllo "HScrollHide" (nascondi barra di scorrimento orizzontale) .

```
virtual bool  OnHScrollHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnScrollLineDown

Il gestore dell'evento (event handler) "ScrollLineDown" (linea di scorrimento verticale verso il basso) .

```
virtual bool  OnScrollLineDown()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnScrollLineUp

Il gestore dell'evento (event handler) "ScrollLineUp" (linea di scorrimento verticale verso l'alto) .

```
virtual bool  OnScrollLineUp()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnScrollLineLeft

L'event handler del controllo "ScrollLineLeft" (linea sinistra di scorrimento orizzontale) .

```
virtual bool  OnScrollLineLeft()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

OnScrollLineRight

L'event handler del controllo "ScrollLineRight" (linea destra di scorrimento orizzontale) .

```
virtual bool  OnScrollLineRight()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

Il metodo della classe base non fa nulla e restituisce sempre true.

ReBound

Imposta le nuove coordinate del controllo utilizzando le coordinate della classe CRect.

```
void  ReBound(
   const & CRect  rect      // Classe CRect
   )
```

Valore di ritorno

Nessuno.

CListView

CListView è una classe del controllo complesso ListView (con i controlli dipendenti).

Descrizione

La Classe CListView incapsula le funzionalità del controllo-lista.

Dichiarazione

```
class CListView : public CWndClient
```

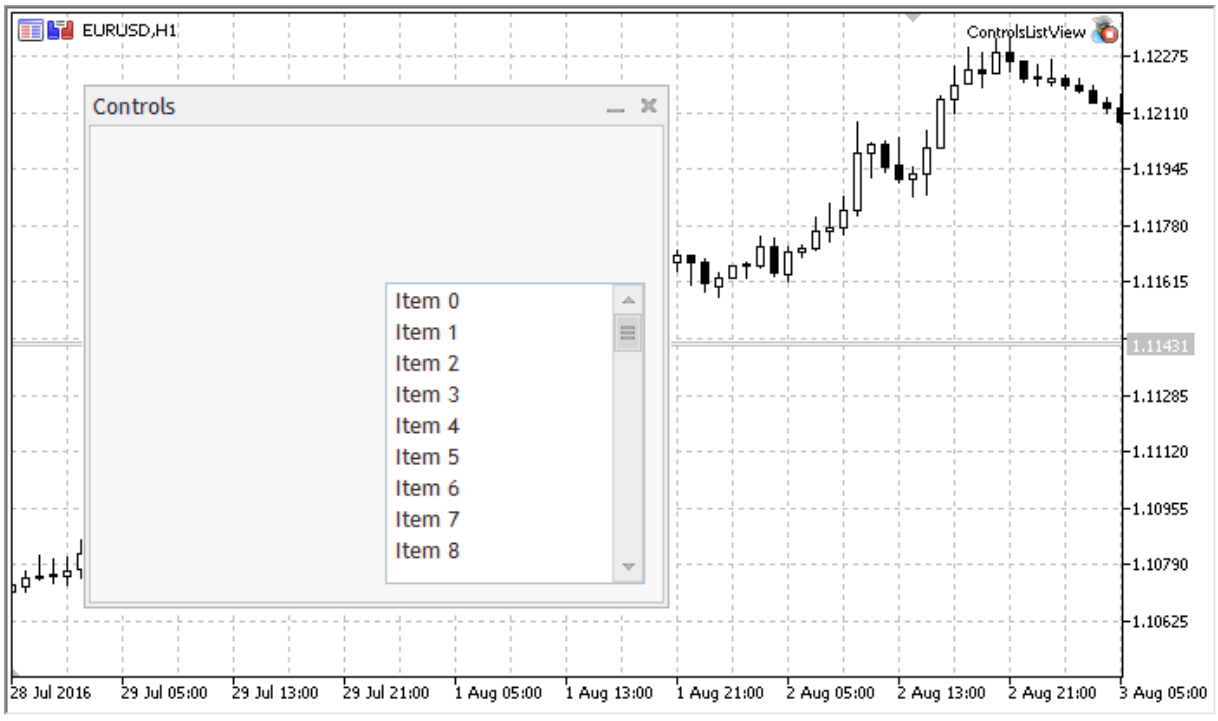
Titolo

```
#include <Controls\ListView.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CWndClient
CListView

Result of the code provided below:



Metodi della Classe

Create	
Create	Crea il controllo
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
Impostazioni	
TotalView	Imposta il numero di elementi, mostrato sul controllo
Aggiungi/Elimina	
AddItem	Aggiunge un elemento
Data	
Select	Seleziona elemento elenco aggiornato in base all'indice
SelectByText	Seleziona elemento della lista corrente per testo
SelectByValue	Seleziona elemento della lista corrente per valore
Di sola lettura dei dati	
Value	Ottiene il valore dell'elemento della lista corrente
Controlli dipendenti	
CreateRow	Crea una fila di ListView
Event handler interni	
OnResize	"Resize" event handler (virtual)
Event handlers dei controlli Dependent	
OnVScrollShow	Event handler "Show" (virtual) del controllo dipendente VScroll
OnVScrollHide	Event handler "Hide" (virtual) del controllo dipendente VScroll
OnScrollLineDown	Event handler "ScrollLineDown" (virtual) del controllo dipendente VScroll
OnScrollLineUp	Event handler "ScrollLineUp" (virtual) del controllo dipendente VScroll
OnItemClick	"ItemClick" event handler (virtual)
Redraw	
Redraw	Ridisegna il controllo
RowState	Imposta lo stato della riga specificata
CheckView	Controlla la "visibilità" della riga specificata

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Metodi ereditati dalla classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Example of creating a panel with list view control:

```
//+------------------------------------------------------------------+
//|                                              ControlsListView.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CListView"
#include <Controls\Dialog.mqh>
#include <Controls\ListView.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
```

```
#define RADIO_HEIGHT                            (56)      // size by Y coordinate
#define CHECK_HEIGHT                            (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CListView         m_list_view;                     // CListView object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateListView(void);
   //--- handlers of the dependent controls events
   void              OnChangeListView(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_list_view,OnChangeListView)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
```

```
      return(false);
//--- create dependent controls
   if(!CreateListView())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "ListView" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateListView(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+2*CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+2*CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+LIST_HEIGHT-CONTROLS_GAP_Y;
//--- create
   if(!m_list_view.Create(m_chart_id,m_name+"ListView",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_list_view))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_list_view.AddItem("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeListView(void)
  {
   Comment(__FUNCTION__+" \""+m_list_view.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
```

```
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CListView.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

TotalView

Imposta il numero di elementi, mostrato sul controllo.

```
bool  TotalView(
   const int  value      // elementi mostrati
   )
```

Parametri

value

[in] Il numero di elementi, mostrati sul controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

Nota

Il numero di elementi mostrati può essere specificato solo in una sola volta.

AddItem

Aggiunge un elemento al controllo.

```
bool  AddItem(
   const string  item,     // testo
   const long    value     // valore
   )
```

Parametri

item

[in] Text.

value

[in] Valore di tipo long.

Valore di ritorno

true in caso di successo, altrimenti false.

Select

Seleziona elemento della lista corrente in base all'indice.

```
bool  Select(
   const int  index     // indice
   )
```

Parametri

index

[in] Indice dell'elemento.

Valore di ritorno

true in caso di successo, altrimenti false.

SelectByText

Seleziona elemento della corrente lista per testo.

```
bool  SelectByText(
   const string  text     // testo
   )
```

Parametri

text

[in] Text.

Valore di ritorno

true in caso di successo, altrimenti false.

SelectByValue

Seleziona elemento della lista corrente per valore.

```
bool  SelectByValue(
   const long  value     // valore
   )
```

Parametri

value

[in] Valore di tipo long.

Valore di ritorno

true in caso di successo, altrimenti false.

Value

Ottiene il valore dell'elemento della lista corrente.

```
long  Value()
```

Valore di ritorno

Il valore dell'elemento della lista corrente.

CreateRow

Crea una riga del controllo CListView.

```
bool  CreateRow(
   const int  index     // indice
   )
```

Parametri

index

[in] Indice dell'elemento.

Valore di ritorno

true in caso di successo, altrimenti false.

OnResize

L'event handler del controllo "Resize"(ridimensiona) .

```
virtual bool  OnResize()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnVScrollShow

Il gestore dell'evento (event handler) "VScrollShow" (mostrare la barra di scorrimento) .

```
virtual bool  OnVScrollShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnVScrollHide

Il gestore dell'evento (event handler) "VScrollHide" (nascondere la barra di scorrimento) .

```
virtual bool  OnVScrollHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnScrollLineDown

Il gestore dell'evento (event handler) "ScrollLineDown" (linea di scorrimento verticale verso il basso) .

```
virtual bool  OnScrollLineDown()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnScrollLineUp

Il gestore dell'evento (event handler) "ScrollLineUp" (linea di scorrimento verticale verso l'alto) .

```
virtual bool  OnScrollLineUp()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnItemClick

L'event handler del controllo "ItemClick" (clic del mouse su una riga).

```
virtual bool  OnItemClick()
   const int    index     // indice della riga
   )
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Redraw

Ridisegna il controllo.

```
bool  Redraw()
```

Valore di ritorno

true in caso di successo, altrimenti false.

RowState

Imposta lo stato della riga specificata.

```
bool  RowState(
   const int   index      // indice
   const bool  select     // stato
   )
```

Parametri

index

[in] Indice della riga.

select

[in] Stato della riga.

Valore di ritorno

true in caso di successo, altrimenti false.

CheckView

Controlla la "visibilità" della riga specificata.

```
bool  CheckView()
```

Valore di ritorno

true, se la riga selezionata è visibile, altrimenti false.

CComboBox

CComboBox è una classe del controllo complesso ComboBox (con i controlli dipendenti).

Descrizione

ComboBox è costituito da un box elementi, combinato con un controllo statico, destinato alla selezione. La porzione lista-box del controllo può essere discesa quando l'utente seleziona la freccia a discesa accanto al controllo

Dichiarazione

```
class CComboBox : public CWndContainer
```

Titolo

```
#include <Controls\ComboBox.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CComboBox

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
Add	
AddItem	Aggiunge un elemento
Impostazioni	
ListViewItems	Imposta il numero di elementi, mostrato sul controllo
Data	
Select	Seleziona elemento elenco aggiornato in base all'indice
SelectByText	Seleziona elemento della lista corrente per testo
SelectByValue	Seleziona elemento della lista corrente per valore
Di sola lettura dei dati	
Value	Ottiene il valore dell'elemento della lista corrente
Controlli dipendenti	
CreateEdit	Crea il controllo dipendente (edit)
CreateButton	Crea il controllo dipendente (bottone)
CreateList	Crea il controllo dipendente (visualizzazione elenco)
Event handlers dei controlli Dependent	
OnClickEdit	"ClickEdit" event handler (virtual)
OnClickButton	"ClickButton" event handler (virtual)
OnChangeList	"ChangeList" event handler (virtual)
Show/Hide	
ListShow	Mostra la lista di elementi
ListHide	Nasconde la lista di elementi

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet,

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

Example of creating a panel with Combobox control:

```
//+------------------------------------------------------------------+
//|                                                ControlsComboBox.mq5 |
//|                        Copyright 2015, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CComboBox"
#include <Controls\Dialog.mqh>
#include <Controls\ComboBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
```

```
private:
   CComboBox         m_combo_box;;                    // CComboBox object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateComboBox(void);
   //--- handlers of the dependent controls events
   void              OnChangeComboBox(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_combo_box,OnChangeComboBox)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateComboBox())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
```

```
//| Create the "ComboBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateComboBox(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_combo_box.Create(m_chart_id,m_name+"ComboBox",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_combo_box))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_combo_box.ItemAdd("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeComboBox(void)
  {
   Comment(__FUNCTION__+" \""+m_combo_box.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CComboBox.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

AddItem

Aggiunge un elemento al controllo.

```
bool  AddItem(
   const string  item,      // testo
   const long    value      // valore
   )
```

Parametri

item

[in] Text.

value=0

[in] Valore di tipo long.

Valore di ritorno

true in caso di successo, altrimenti false.

ListViewItems

Imposta il numero di elementi della lista del controllo CComboBox.

```
void  ListViewItems(
   const int    value     // numero di elementi della lista
   )
```

Parametri

value

[in] Numero di elementi della lista.

Valore di ritorno

true in caso di successo, altrimenti false.

Select

Seleziona elemento della lista corrente in base all'indice.

```
bool  Select(
   const int  index     // indice
   )
```

Parametri

index

[in] Indice dell'elemento.

Valore di ritorno

true in caso di successo, altrimenti false.

SelectByText

Seleziona elemento della corrente lista per testo.

```
bool  SelectByText(
   const string  text     // testo
   )
```

Parametri

text

[in] Testo dell'elemento.

Valore di ritorno

true in caso di successo, altrimenti false.

SelectByValue

Seleziona elemento della lista corrente per valore.

```
bool  SelectByValue(
   const long  value     // valore
   )
```

Parametri

value

[in] Valore di tipo long.

Valore di ritorno

true in caso di successo, altrimenti false.

Value

Ottiene il valore dell'elemento della lista corrente.

```
long  Value()
```

Valore di ritorno

Il valore dell'elemento della lista corrente.

CreateEdit

Crea il controllo dipendente (edit) del controllo.

```
virtual bool  CreateEdit()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateButton

Crea il controllo dipendente (bottone).

```
virtual bool  CreateButton()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateList

Crea il controllo dipendente (visualizzazione elenco).

```
virtual bool  CreateList()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnClickEdit

L'event handler del controllo "ClickEdit" (clic del mouse su edit(modifica)).

```
virtual bool  OnClickEdit()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnClickButton

L'event handler del controllo "ClickButton" (clic del mouse sul bottone).

```
virtual bool  OnClickButton()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChangeList

L'event handler "ChangeList" (cambio della lista).

```
virtual bool  OnChangeList()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

ListShow

Mostra la lista elementi.

```
virtual bool  ListShow()
```

Valore di ritorno

true in caso di successo, altrimenti false.

ListHide

Nasconde la lista elementi.

```
virtual bool  ListHide()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CCheckBox

CCheckBox è una classe del controllo complesso CheckBox.

Descrizione

Il Controllo CCheckBox visualizza una casella di controllo che consente all'utente di selezionare una condizione vera o falsa.

Dichiarazione

```
class CCheckBox : public CWndContainer
```

Titolo

```
#include <Controls\CheckBox.mqh>
```

Gerarchia di ereditarietà

CObject
  CWnd
    CWndContainer
      CCheckBox

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo

Create	
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
Proprietà	
Testo	Ottiene/Imposta l'etichetta di testo, associata al controllo
Color	Ottiene/Imposta il colore dell' etichetta di testo, associato al controllo
State	
Checked	Ottiene/Imposta il valore che indica se il controllo viene controllato
Data	
Value	Ottiene/Imposta il valore associato al controllo
Controlli dipendenti	
CreateButton	Crea il controllo dipendente (bottone)
CreateLabel	Crea il controllo dipendente (etichetta)
Event handlers dei controlli Dependent	
ClickButton	"ClickButton" event handler (virtual)
ClickLabel	"ClickLabel" event handler (virtual)

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Example of creating a panel with Checkbox control:

```
//+------------------------------------------------------------------+
//|                                              ControlsCheckBox.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckBox"
#include <Controls\Dialog.mqh>
#include <Controls\CheckBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allow
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckBox         m_check_box1;                    // CCheckBox object
   CCheckBox         m_check_box2;                    // CCheckBox object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- create dependent controls
   bool              CreateCheckBox1(void);
   bool              CreateCheckBox2(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckBox1(void);
```

```
   void              OnChangeCheckBox2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_box1,OnChangeCheckBox1)
ON_EVENT(ON_CHANGE,m_check_box2,OnChangeCheckBox2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckBox1())
      return(false);
   if(!CreateCheckBox2())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
```

```
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box1.Create(m_chart_id,m_name+"CheckBox1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box1.Text("CheckBox1"))
      return(false);
   if(!m_check_box1.Color(clrBlue))
      return(false);
   if(!Add(m_check_box1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box2.Create(m_chart_id,m_name+"CheckBox2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box2.Text("CheckBox2"))
      return(false);
   if(!m_check_box2.Color(clrBlue))
      return(false);
   if(!Add(m_check_box2))
      return(false);
   m_check_box2.Checked(true);
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox1(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box1.Checked()));
  }
//+------------------------------------------------------------------+
```

```
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox2(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CCheckBox.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

Text (Metodo Get)

Ottiene il testo dell'etichetta, associata al controllo.

```
string  Text()
```

Valore di ritorno

Testo dell'etichetta.

Text (Metodo Set)

Imposta il testo dell'etichetta, associata al controllo.

```
bool  Text(
   const string  value      // testo
   )
```

Parametri

value

[in] Nuovo valore dell'etichetta.

Valore di ritorno

true in caso di successo, altrimenti false.

Color (Metodo Get)

Ottiene il colore dell'etichetta, associata al controllo.

```
color  Color()  const
```

Valore di ritorno

Colore dell'etichetta.

Color (Metodo Set)

Imposta il colore dell'etichetta, associata al controllo.

```
bool  Color(
   const color  value      // colore
   )
```

Parametri

value

[in] Nuovo colore dell'etichetta.

Valore di ritorno

true in caso di successo, altrimenti false.

Checked (Metodo Get)

Ottiene stato del controllo.

```
bool  Checked()  const
```

Valore di ritorno

Stato del controllo.

Checked (Metodo Set)

Imposta lo stato del controllo.

```
bool  Checked(
   const bool  flag      // stato
   )
```

Parametri

flag

[in] Nuovo stato.

Valore di ritorno

true in caso di successo, altrimenti false.

Value (Metodo Get)

Ottiene il valore, associato al controllo.

```
int  Value()  const
```

Valore di ritorno

Il valore, associato al controllo.

Value (Metodo Set)

Imposta il valore, associato al controllo.

```
void  Value(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo Valore.

Valore di ritorno

Nessuno.

CreateButton

Crea il controllo dipendente (bottone).

```
virtual bool  CreateButton()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateLabel

Crea il controllo dipendente (etichetta).

```
virtual bool  CreateLabel()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnClickButton

L'event handler del controllo "ClickButton" (clic del mouse sul bottone).

```
virtual bool  OnClickButton()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnClickLabel

L'event handler del controllo "ClickLabel" (clic del mouse sull'etichetta).

```
virtual bool  OnClickLabel()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CCheckGroup

CCheckGroup è una classe di controllo complesso CheckGroup (con i controlli dipendenti).

Descrizione

CCheckGroup fornisce la possibilità per la creazione di controlli, che consentono di visualizzare e modificare le flags.

Dichiarazione

```
class CCheckGroup : public CWndClient
```

Titolo

```
#include <Controls\CheckGroup.mqh>
```

Gerarchia di ereditarietà

CObject
  CWnd
    CWndContainer
      CWndClient
        CCheckGroup

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo
Event handlers chart	
OnEvent	Event handler per per tutti gli eventi chart
Add	
AddItem	Aggiunge un nuovo elemento
Di sola lettura dei dati	
Value	Ottiene il valore, associato al controllo
Controlli dipendenti	
CreateButton	Crea nuovo elemento CCheckBox
Event handlers dei controlli Dependent	
OnVScrollShow	Event handler "Show" (virtual) del controllo dipendente VScroll
OnVScrollHide	Event handler "Hide" (virtual) del controllo dipendente VScroll
OnScrollLineDown	Event handler "ScrollLineUp" (virtual) del controllo dipendente VScroll
OnScrollLineUp	Event handler "ScrollLineDown" (virtual) del controllo dipendente VScroll
OnChangeItem	Event handler "ChangeItem" (virtual) del controllo dipendente VScroll
Redraw	
Redraw	Ridisegna il gruppo
RowState	Imposta lo stato dell'elemento specificato

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Example of creating a panel with Checkbox group control:

```
//+------------------------------------------------------------------+
//|                                            ControlsCheckGroup.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckGroup"
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckGroup       m_check_group;                  // CCheckGroup object
```

```
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateCheckGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckGroup())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckGroup" element                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
```

```
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+CHECK_HEIGHT;
//--- create
   if(!m_check_group.Create(m_chart_id,m_name+"CheckGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_check_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<5;i++)
      if(!m_check_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_check_group.Check(0,1<<0);
   m_check_group.Check(2,1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
```

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea nuovo controllo CCheckGroup.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

AddItem

Aggiunge un elemento al controllo.

```
bool  AddItem(
   const string  item,      // testo
   const long    value      // valore
   )
```

Parametri

item

[in] Text.

value=0

[in] Valore di tipo long.

Valore di ritorno

true in caso di successo, altrimenti false.

Value

Ottiene il valore, associato al controllo.

```
long  Value()
```

Valore di ritorno

Il valore associato al controllo.

Nota

Il valore dipende dallo stato di tutti gli elementi del CCheckGroup.

CreateButton

Crea nuova istanza della classe CCheckBox sull'indice specificato.

```
bool  CreateButton(
   int  index      // indice
   )
```

Parametri

index

[in] Indice del nuovo elemento nel CCheckGroup.

Valore di ritorno

true in caso di successo, altrimenti false.

OnVScrollShow

Il gestore dell'evento (event handler) "VScrollShow" (mostrare la barra di scorrimento) .

```
virtual bool  OnVScrollShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnVScrollHide

Il gestore dell'evento (event handler) "VScrollHide" (nascondere la barra di scorrimento) .

```
virtual bool  OnVScrollHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnScrollLineDown

Il gestore dell'evento (event handler) "ScrollLineDown" (linea di scorrimento verticale verso il basso) .

```
virtual bool  OnScrollLineDown()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnScrollLineUp

Il gestore dell'evento (event handler) "ScrollLineUp" (linea di scorrimento verticale verso l'alto) .

```
virtual bool  OnScrollLineUp()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChangeItem

L'event handler del controllo "ChangeItem" (cambio elemento).

```
virtual bool  OnChangeItem(
   const int  index      // indice(index)
   )
```

Parametri

index

[in] indice dell'elemento modificato.

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Redraw

Ridisegna il controllo.

```
bool  Redraw()
```

Valore di ritorno

true in caso di successo, altrimenti false.

RowState

Imposta lo stato dell'elemento specificato.

```
bool  RowState(
   const int   index,      // indice dell'elemento
   const bool  select      // stato
   )
```

Parametri

index

[in] Indice dell'elemento da modificare.

select

[in] Nuovo stato.

Valore di ritorno

true in caso di successo, altrimenti false.

CRadioButton

CRadioButton è una classe dei complessi controlli di RadioButton.

Descrizione

CRadioButton in se non è utilizzato; è utilizzato per la creazione di elementi CRadioGroup.

Dichiarazione

```
class CRadioButton : public CWndContainer
```

Titolo

```
#include <Controls\RadioButton.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CRadioButton

Metodi della Classe

Create	
Create	Crea il controllo
Event handlers chart	
OnEvent	Event handler per per tutti gli eventi chart
Proprietà	
Testo	Ottiene/Imposta l'etichetta di testo, associata con il controllo
Color	Ottiene/Imposta il colore di etichetta di testo, associato al controllo
State	
State	Ottiene/Imposta lo stato
Controlli dipendenti	
CreateButton	Crea botoone
CreateLabel	Crea etichetta
Event handlers dei controlli Dependent	
OnClickButton	"ClickButton" event handler (virtual)
OnClickLabel	"ClickLabel" event handler (virtual)

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Create

Crea nuovo controllo CRadioButton.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

Text (Metodo Get)

Ottiene il testo dell'etichetta, associata al controllo.

```
string  Text()
```

Valore di ritorno

Testo dell'etichetta.

Text (Metodo Set)

Imposta il testo dell'etichetta, associata al controllo.

```
bool  Text(
   const string  value      // testo
   )
```

Parametri

value

[in] Nuovo valore dell'etichetta.

Valore di ritorno

true in caso di successo, altrimenti false.

Color (Metodo Get)

Ottiene il colore dell'etichetta, associata al controllo.

```
color  Color()  const
```

Valore di ritorno

Colore dell'etichetta.

Color (Metodo Set)

Imposta il colore dell'etichetta, associata al controllo.

```
bool  Color(
   const color  value      // colore
   )
```

Parametri

value

[in] Nuovo colore dell'etichetta.

Valore di ritorno

true in caso di successo, altrimenti false.

State (Metodo Get)

Ottiene lo stato del bottone.

```
bool  State()  const
```

Valore di ritorno

Stato del Bottone.

State (Metoto Set)

Imposta lo stato del Bottone.

```
bool  State(
   const bool  flag      // flag
   )
```

Parametri

flag

[in] Nuovo stato del Bottone.

Valore di ritorno

true in caso di successo, altrimenti false.

CreateButton

Crea Bottone.

```
virtual bool  CreateButton()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateLabel

Crea etichetta.

```
virtual bool  CreateLabel()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnClickButton

L'event handler del controllo "ClickButton" (clic del mouse).

```
virtual bool  OnClickButton()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnClickLabel

L'event handler del controllo "ClickLabel" (clic del mouse sull'etichetta).

```
virtual bool  OnClickLabel()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CRadioGroup

CRadioGroup è di classe del controllo complesso RadioGroup (con i controlli dipendenti).

Descrizione

CRadioGroup consente all'utente di selezionare una singola opzione da un gruppo di scelte quando accoppiato con altri controlli CRadioButton.

Dichiarazione

```
class CRadioGroup : public CWndClient
```

Titolo

```
#include <Controls\RadioGroup.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CWndClient
CRadioGroup

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
Add	
AddItem	Aggiunge un nuovo elemento
Di sola lettura dei dati	
Value	Ottiene il valore, associato al controllo
Controlli dipendenti	
CreateButton	Crea nuovo elemento CRadioButton
Event handlers dei controlli Dependent	
OnVScrollShow	Event handler "Show" (virtual) del controllo dipendente VScroll
OnVScrollHide	Event handler "Hide" (virtual) del controllo dipendente VScroll
OnScrollLineDown	Event handler "ScrollLineDown" (virtual) del controllo dipendente VScroll
OnScrollLineUp	Event handler "ScrollLineUp" (virtual) del controllo dipendente VScroll
OnChangeItem	"ChangeItem" event handler (virtual)
Redraw	
Redraw	Ridisegna gli elementi del gruppo
RowState	Imposta lo stato dell'elemento specificato
Select	Seleziona elemento corrente

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Example of creating a panel with group of radio buttons:

```
//+------------------------------------------------------------------+
//|                                            ControlsRadioGroup.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CRadioGroup"
#include <Controls\Dialog.mqh>
#include <Controls\RadioGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CRadioGroup       m_radio_group;                  // CRadioGroup object
```

```
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateRadioGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeRadioGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_radio_group,OnChangeRadioGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateRadioGroup())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "RadioGroup" element                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateRadioGroup(void)
```

```
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+RADIO_HEIGHT;
//--- create
   if(!m_radio_group.Create(m_chart_id,m_name+"RadioGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_radio_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<3;i++)
      if(!m_radio_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_radio_group.Value(1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeRadioGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Crea un nuovo controllo CRadioGroup.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

AddItem

Aggiunge un elemento al controllo.

```
bool  AddItem(
   const string  item,      // testo
   const long    value=0    // valore
   )
```

Parametri

item

[in] Text.

value=0

[in] Valore di tipo long.

Valore di ritorno

true in caso di successo, altrimenti false.

Value

Ottiene il valore, associato al controllo.

```
long  Value()
```

Valore di ritorno

Ilvalore associato al controllo.

Nota

Il valore dipende dallo stato di tutti gli elementi CRadioButton del controllo CRadioGroup.

CreateButton

Crea nuova istanza della classe CRadioButton sull'indice specificato.

```
bool  CreateButton(
   const int  index      // indice(index)
   )
```

Parametri

index

[in] Indice del nuovo elemento nel CRadioGroup.

Valore di ritorno

true in caso di successo, altrimenti false.

OnVScrollShow

Il gestore dell'evento (event handler) "VScrollShow" (mostrare la barra di scorrimento) .

```
virtual bool  OnVScrollShow()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnVScrollHide

Il gestore dell'evento (event handler) "VScrollHide" (nascondere la barra di scorrimento) .

```
virtual bool  OnVScrollHide()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnScrollLineDown

Il gestore dell'evento (event handler) "ScrollLineDown" (linea di scorrimento verticale verso il basso) .

```
virtual bool  OnScrollLineDown()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnScrollLineUp

Il gestore dell'evento (event handler) "ScrollLineUp" (linea di scorrimento verticale verso l'alto) .

```
virtual bool  OnScrollLineUp()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChangeItem

L'event handler del controllo "ChangeItem" (cambio elemento).

```
virtual bool  OnChangeItem(
   const int  index      // indice(index)
   )
```

Parametri

index

[in] indice dell'elemento modificato.

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Redraw

Ridisegna il controllo.

```
bool  Redraw()
```

Valore di ritorno

true in caso di successo, altrimenti false.

RowState

Imposta lo stato dell'elemento specificato.

```
bool  RowState(
   const int   index,      // indice dell'elemento
   const bool  select      // stato
   )
```

Parametri

index

[in] Indice dell'elemento da modificare.

select

[in] Nuovo stato.

Valore di ritorno

true in caso di successo, altrimenti false.

Select

Seleziona elemento corrente.

```
void  Select(
   const int  index      // indice(index)
   )
```

Parametri

index

[in] Indice dell'elemento da selezionare.

Valore di ritorno

Nessuno.

CSpinEdit

CSpinEdit è una classe del controllo complesso SpinEdit controllo (con i controlli dipendenti).

Descrizione

La Classe CSpinEdit è intesa per la creazione di controlli che permettono di modificare i valori di tipo intero(integer). Aumenta automaticamente i dati quando si preme il pulsante in alto, e diminuisce se si preme il pulsante in basso.

Dichiarazione

```
class CSpinEdit : public CWndContainer
```

Titolo

```
#include <Controls\SpinEdit.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CSpinEdit

Result of the code provided below:

Metodi della Classe

Create	
Create	Crea il controllo
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
Proprietà	
MinValue	Ottiene/Imposta il valore minimo consentito
MaxValue	Ottiene/Imposta il valore massimo consentito
State	
Value	Ottiene/Imposta il valore corrente
Controlli dipendenti	
CreateEdit	Crea il controllo dipendente (edit)
CreateInc	Crea il controllo dipendente (bottone di incremento)
CreateDec	Crea il controllo dipendente (bottone di decremento)
Event handlers dei controlli Dependent	
OnClickInc	"ClickInc" event handler (virtual)
OnClickDec	"ClickDec" event handler (virtual)
Event handler interni	
OnChangeValue	"ChangeValue" event handler (virtual)

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Example of creating a panel with spin edit control:

```
//+------------------------------------------------------------------+
//|                                                ControlsSpinEdit.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
```

```
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CSpinEdit"
#include <Controls\Dialog.mqh>
#include <Controls\SpinEdit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CSpinEdit         m_spin_edit;                    // CSpinEdit object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateSpinEdit(void);
   //--- handlers of the dependent controls events
```

```
   void              OnChangeSpinEdit(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
   ON_EVENT(ON_CHANGE,m_spin_edit,OnChangeSpinEdit)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateSpinEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "SpinEdit" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateSpinEdit(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+(BUTTON_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_spin_edit.Create(m_chart_id,m_name+"SpinEdit",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_spin_edit))
      return(false);
   m_spin_edit.MinValue(10);
```

```
   m_spin_edit.MaxValue(100);
   m_spin_edit.Value(50);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeSpinEdit(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- clear comments
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
```

```
}
```

Create

Crea nuovo controllo CSpinEdit.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

MinValue (Metodo Get)

Ottiene il valore della proprietà "MinValue" (valore consentito minimo) del controllo.

```
int  MinValue()  const
```

Valore di ritorno

Il valore della proprietà "MinValue".

MinValue (Metodo Set)

Imposta il valore della proprietà "MinValue" (valore consentito minimo) del controllo.

```
void  MinValue(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà "MinValue".

Valore di ritorno

Nessuno.

MaxValue (Metodo Get)

Ottiene il valore della proprietà "MaxValue" (valore massimo consentito) del controllo.

```
int  MaxValue()  const
```

Valore di ritorno

Il valore della proprietà "MaxValue".

MaxValue (Metoto Set)

Imposta il valore della proprietà "MaxValue" (valore massimo consentito) del controllo.

```
void  MaxValue(
   const int  value      // nuovo valore
   )
```

Parametri

value

[in] Nuovo valore della proprietà "MaxValue".

Valore di ritorno

Nessuno.

Value (Metodo Get)

Ottiene la proprietà "Value" (valore corrente) del controllo.

```
int  Value()  const
```

Valore di ritorno

La proprietà "Value" (valore).

Value (Metodo Set)

Imposta la proprietà "Value" (valore corrente) del controllo.

```
void  Value(
   const int  value      // valore
   )
```

Parametri

value

[in] Nuovo "Valore"(Value) della proprietà.

Valore di ritorno

Nessuno.

CreateEdit

Crea il controllo dipendente (CEdit).

```
virtual bool  CreateEdit()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateInc

Crea il controllo dipendente (bottone di incremento).

```
virtual bool  CreateInc()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateDec

Crea il controllo dipendente (bottone di decremento).

```
virtual bool  CreateDec()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnClickInc

L'event handler del controllo "ClickInc" (click tdel mouse sul bottone incremento).

```
virtual bool  OnClickInc()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnClickDec

L'event handler del controllo "ClickDec" (click del mouse sul bottone decremento).

```
virtual bool  OnClickDec()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

OnChangeValue

L'event handler del controllo "ChangeValue" .

```
virtual bool  OnChangeValue()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

CDialog

CDialog è una classe del controllo complesso Dialog(finestra dialogo).

Descrizione

La classe CDialog è destinata a combinare i controlli con differenti funzioni del gruppo.

Dichiarazione

```
class CDialog : public CWndContainer
```

Titolo

```
#include <Controls\Dialog.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CDialog

Discendenti diretti

CAppDialog

Metodi della Classe

Create	
Create	Crea il controllo
Event handlers chart	
OnEvent	Event Handler di tutti gli eventi del chart
Proprietà	
Didascalia	Ottiene/Imposta il valore della proprietà "Caption"(didascalia)
Add	
Add	Aggiunge il controllo per l'area client
Controlli dipendenti	
CreateWhiteBorder	Crea il controllo dipendente (bordo bianco)
CreateBackground	Crea il controllo dipendente (sfondo)
CreateCaption	Crea il controllo dipendente (didacalia)
CreateButtonClose	Crea il controllo dipendente (bottone chiudi)
CreateClientArea	Crea il controllo dipendente (area client)

Create	
Event handlers dei controlli Dependent	
OnClickCaption	"ClickCaption" event handler
OnClickButtonClose	"ClickButtonClose" event handler
L'accesso all'area client	
ClientAreaVisible	Imposta un valore che indica se l'area client è visibile
ClientAreaLeft	Ottiene la coordinata X dell'angolo superiore sinistro dell'area client del controllo
ClientAreaTop	Ottiene la coordinata Y dell'angolo superiore sinistro dell'area client del controllo
ClientAreaRight	Ottiene la coordinata X dell'angolo inferiore destro dell'area client del controllo
ClientAreaBottom	Ottiene la coordinata Y dell'angolo inferiore destro dell'area client del controllo
ClientAreaWidth	Ottiene la larghezza dell'area client
ClientAreaHeight	Ottiene l'altezza dell'area client
Event handlers trascinamento	
OnDialogDragStart	"DialogDragStart" event handler (virtual)
OnDialogDragProcess	"DialogDragProcess" event handler (virtual)
OnDialogDragEnd	"DialogDragEnd" event handler (virtual)

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Create

Crea nuovo controllo CDialog.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

Caption (Metodo Get)

Ottiene la proprietà "Caption" del controllo CDialog.

```
string  MinValue()  const
```

Valore di ritorno

La proprietà "Caption".

Caption (Metodod Set)

Imposta la proprietà "Caption" del controllo CDialog.

```
bool  Caption(
   const string  text      // testo
   )
```

Parametri

text

[in] Nuovo valore della proprietà "Caption".

Valore di ritorno

true in caso di successo, altrimenti false.

Add

Aggiunge il controllo all'area client per puntatore.

```
bool  Add(
   CWnd   *control,         // puntatore
   )
```

Parametri

control

[in] Puntatore al controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

Add

Aggiunge il controllo per l'area client per riferimento.

```
bool  Add(
   CWnd   &control,         // riferimento
   )
```

Parametri

control

[in] Riferimento al controllo.

Valore di ritorno

true in caso di successo, altrimenti false.

CreateWhiteBorder

Crea il controllo dipendente (bordo bianco).

```
virtual bool  CreateWhiteBorder()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateBackground

Crea il controllo dipendente (sfondo).

```
virtual bool  CreateBackground()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateCaption

Crea il controllo dipendente (caption).

```
virtual bool  CreateCaption()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateButtonClose

Crea il controllo dipendente (bottone chiudi)

```
virtual bool  CreateButtonClose()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateClientArea

Crea il controllo dipendente (area client).

```
virtual bool  CreateClientArea()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnClickCaption

L'event handler del controllo "ClickCaption".

```
virtual bool  OnClickCaption()
```

Valore di ritorno

true in caso di successo, altrimenti false.

OnClickButtonClose

L'event handler del controllo "ClickButtonClose".

```
virtual bool  OnClickButtonClose()
```

Valore di ritorno

true in caso di successo, altrimenti false.

ClientAreaVisible

Imposta un flag che indica se l'area client è visibile.

```
bool  ClientAreaVisible(
   const bool  visible      // flag di visibilità
   )
```

Parametri

visible

[in] Flag di visibilità.

Valore di ritorno

true in caso di successo, altrimenti false.

ClientAreaLeft

Ottiene la coordinata X dell'angolo superiore sinistro del controllo dell'area client.

```
int  ClientAreaLeft()
```

Valore di ritorno

La coordinata X del controllo dell angolo dell'area client in alto a sinistra.

ClientAreaTop

Ottiene la coordinata Y dell'angolo superiore sinistro dell'area client del controllo

```
int  ClientAreaTop()
```

Valore di ritorno

La coordinata Y del controllo dell'angolo superiore sinistro dell'area client.

ClientAreaRight

Ottiene la coordinata X dell'angolo inferiore destro del controllo dell'area client.

```
int  ClientAreaTop()
```

Valore di ritorno

La coordinata X nell'angolo in basso a destra del controllo dell'area client.

ClientAreaBottom

Ottiene la coordinata Y nell'angolo in basso a destra del controllo dell'area client.

```
int  ClientAreaBottom()
```

Valore di ritorno

La coordinata Y nell'angolo in basso a destra del controllo dell'area client.

ClientAreaWidth

Ottiene la larghezza del controllo dell'area client.

```
int  ClientAreaWidth()
```

Valore di ritorno

La larghezza del controllo dell'area client.

ClientAreaHeight

Ottiene l'altezza del controllo dell dell'area client.

```
int  ClientAreaHeight()
```

Valore di ritorno

L'altezza del controllo dell'area client.

OnDialogDragStart

L'event handler del controllo "DialogDragStart" .

```
virtual bool  OnDialogDragStart()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "DialogDragStart" si verifica all'inizio del trascinamento del controllo.

OnDialogDragProcess

L'event handler del controllo "DialogDragProcess" .

```
virtual bool  OnDialogDragProcess()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "DialogDragProcess" si verifica quando il controllo viene trascinato.

OnDialogDragEnd

L'event handler del controllo "Hide" .

```
virtual bool  OnDialogDragEnd()
```

Valore di ritorno

true se l'evento è stato elaborato, altrimenti false.

Nota

L'evento "DialogDragEnd" si verifica alla fine del trascinamento del controllo.

CAppDialog

CAppDialog è una classe di Applocazione Finestra di dialogo di controllo complesso (con i controlli dipendenti).

Descrizione

La classe CAppDialog è destinata a combinare i controlli con funzioni diverse nel gruppo all'interno del programma MQL5.

Dichiarazione

```
class CAppDialog : public CDialog
```

Titolo

```
#include <Controls\Dialog.mqh>
```

Gerarchia di ereditarietà

CObject
CWnd
CWndContainer
CDialog
CAppDialog

Metodi della Classe

Creare e distruggere	
Create	Crea il controllo
Destroy	Distrugge controllo
Elaborazione eventi	
OnEvent	Event Handler di tutti gli eventi del chart
Run	
Run	Esegue controllo
Elaborazione eventi del Chart	
ChartEvent	Event Handler di tutti gli eventi del chart
Impostazioni	
Minimized	Imposta un valore che indica se il controllo è minimizzato
Salva/Carica	
IniFileSave	Salva lo stato di controllo del file

Creare e distruggere	
IniFileLoad	Carica lo stato di controllo dal file
IniFileName	Imposta il nome del file per il caricamento/salvataggio dello stato di controllo
IniFileExt	Imposta l'estensione del file per il caricamento/salvataggio dello stato di controllo
Inizializzazione	
CreateCommon	Metodo di inizializzazione Comune
CreateExpert	Metodo di inizializzazione per lavorare con Expert Advisors
CreateIndicator	Metodo di inizializzazione per lavorare con indicatori
Controlli dipendenti	
CreateButtonMinMax	Crea controlli dipendenti (minimizza/massimizza bottoni)
Event handlers dei controlli Dependent	
OnClickButtonClose	"ClickButtonClose" event handler (virtual)
OnClickButtonMinMax	"ClickButtonMinMax" event handler (virtual)
Eventi esterni	
OnAnotherApplicationClose	Event handler di eventi esterni (virtual)
Methods	
Rebound	Imposta le nuove coordinate del controllo utilizzando le coordinate della classe CRect
Minimize	Mostra il controllo nello stato minimizzato
Maximize	Mostra il controllo nello stato ingrandito (ripristinato)
CreateInstanceId	Crea un ID univoco per i nomi degli oggetti di controllo
ProgramName	Ottiene il nome del programma MQL5
SubwinOff	Ottiene l'offset Y della sottofinestra di controllo

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Metodi ereditati dalla classe CObject

Prev, Prev, Next, Next, Type, Compare

Metodi ereditati dalla classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Metodi ereditati dalla classe CDialog

Caption, Caption, Add, Add

Create

Crea nuovo controllo CAppDialog.

```
virtual bool  Create(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

Destroy

Metodo di deinitializzazione del controllo CAppDialog .

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // codice motivazione
)
```

Parametri

reason

[in] Codice motivo di deinitializzazione. REASON_PROGRAM è impostato di default.

Valore di ritorno

Nessuno.

OnEvent

Event Handler del Chart

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

Run

Esegue il controllo.

```
bool  Run()
```

Valore di ritorno

true in caso di successo, altrimenti false.

ChartEvent

Event Handler del Chart

```
virtual bool  ChartEvent(
   const int      id,          // ID
   const long&    lparam,      // parametro evento di tipo long
   const double&  dparam,      // parametro evento di tipo double
   const string&  sparam       // parametro evento di tipo string
   )
```

Parametri

id

[in] Event ID.

lparam

[in] Parametro evento di tipo long passato per riferimento.

dparam

[in] Parametro evento di tipo double passato per riferimento.

sparam

[in] Parametro evento di tipo string passato per riferimento.

Valore di ritorno

true - se l'evento è stato elaborato, altrimenti false.

Minimized

Imposta il valore della proprietà "Minimizzata" del controllo.

```
bool  Minimized(
   const bool  flag      // stato
   )
```

Parametri

flag

[in] Nuovo stato.

Valore di ritorno

true in caso di successo, altrimenti false.

IniFileSave

Salva lo stato di controllo di file.

```
void  IniFileSave()
```

Valore di ritorno

true in caso di successo, altrimenti false.

IniFileLoad

Carica lo stato di controllo da file.

```
void  IniFileLoad()
```

Valore di ritorno

true in caso di successo, altrimenti false.

IniFileName

Imposta il nome del file per il caricamento/salvataggio dello stato di controllo.

```
virtual string  IniFileName()  const
```

Valore di ritorno

Il nome del file per il caricamento/salvataggio dello stato di controllo.

Nota

Il nome del file include il nome dell'Expert Advisor / Indicatore ed il simbolo di lavoro, su cui viene lanciato il programma MQL5.

IniFileExt

Imposta l'estensione del file per il caricamento/salvataggio dello stato di controllo.

```
virtual string  IniFileExt()  const
```

Valore di ritorno

Estensione del file, utilizzata per il caricamento/salvataggio dello stato del controllo.

CreateCommon

Metodi di inizializzazione comune.

```
bool  CreateCommon(
   const long     chart,       // chart ID
   const string   name,        // nome
   const int      subwin,      // sottofinestra chart
   )
```

Parametri

chart

[in] chart ID.

name

[in] Nome univoco del controllo.

subwin

[in] Sottofinestra Chart.

Valore di ritorno

true in caso di successo, altrimenti false.

CreateExpert

Metodo di inizializzazione per lavorare in Expert Advisors.

```
bool  CreateExpert(
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

CreateIndicator

Metodo di inizializzazione per lavorare con indicatori.

```
bool  CreateIndicator(
   const int      x1,          // x1 coordinate
   const int      y1,          // y1 coordinate
   const int      x2,          // x2 coordinate
   const int      y2           // y2 coordinate
   )
```

Parametri

x1

[in] X coordinate dell'angolo superiore sinistro.

y1

[in] Y coordinate dell'angolo superiore sinistro.

x2

[in] X coordinate dell'angolo inferiore destro.

y2

[in] Y coordinate dell'angolo inferiore destro.

Valore di ritorno

true in caso di successo, altrimenti false.

CreateButtonMinMax

Crea controlli dipendenti (minimizzare/massimizzare bottoni).

```
virtual void  CreateButtonMinMax()
```

Valore di ritorno

Nessuno.

OnClickButtonClose

L'event handler del controllo "ClickButtonClose" (clic del mouse sul pulsante di chiusura).

```
virtual void  OnClickButtonClose()
```

Valore di ritorno

Nessuno.

OnClickButtonMinMax

L'event handler del controllo "ClickButtonMinMax" (clic del mouse sul pulsante di massimizza/minimizza).

```
virtual void  OnClickButtonClose()
```

Valore di ritorno

Nessuno.

OnAnotherApplicationClose

Event handler di eventi esterni.

```
virtual void  OnAnotherApplicationClose()
```

Valore di ritorno

Nessuno.

Rebound

Imposta le nuove coordinate del controllo utilizzando le coordinate della classe CRect.

```
bool  Rebound(
   const & CRect  rect      // Classe CRect
   )
```

Valore di ritorno

true in caso di successo, altrimenti false.

Minimize

Mostra il controllo nello stato ridotto al minimo.

```
virtual void  Minimize()
```

Valore di ritorno

true in caso di successo, altrimenti false.

Maximize

Mostrail controllo nello stato ingrandito (ripristinato).

```
virtual void  Maximize()
```

Valore di ritorno

true in caso di successo, altrimenti false.

CreateInstanceId

Crea un ID univoco per i nomi degli oggetti di controllo.

```
string  CreateInstanceId()
```

Valore di ritorno

Prefisso per i nomi degli oggetti.

ProgramName

Ottiene il nome del programma MQL5.

```
string  ProgramName()
```

Valore di ritorno

Nome del programma MQL5.

SubwinOff

Ottiene l'offset Y della finestra secondaria di controllo.

```
void  SubwinOff()
```

Valore di ritorno

Nessuno.

Il passaggio da MQL4 ad MQL5

MQL5 è l'evoluzione del suo predecessore - il linguaggio di programmazione MQL4, in cui sono stati scritti numerosi indicatori, script e Expert Advisor. Nonostante il fatto che il nuovo linguaggio di programmazione sia massimamente compatibile con il linguaggio della precedente generazione, ci sono ancora alcune differenze tra questi linguaggi. E quando si trasferiscono i programmi, queste differenze devono essere notate.

Questa sezione contiene informazioni destinate a facilitare l'adattamento dei codici al nuovo linguaggio MQL5 per i programmatori che conoscono MQL4.

Prima di tutto va rilevato che:

- Il nuovo linguaggo non contiene le funzioni start(), init() e deinit().
- Non ci sono limiti per il numero di buffer di indicatori.
- LE DLL vengono caricate immediatamente dopo il caricamento di un Expert Advisor (o qualsiasi altro programma mql5).
- La verifica delle condizioni logiche è stata abbreviata.
- Quando i limiti di un array vengono superati, la performance corrente viene terminata (criticamente - con l'uscita di un errore).
- Precedenza degli operatori come in C + +.
- Il linguaggio offre il tipo di cast implicito (anche da stringa ad un numero).
- Le variabili locali non vengono inizializzate automaticamente (tranne che per le stringhe).
- Comuni array locali vengono cancellati automaticamente.

Funzioni speciali init, start e deinit

Il linguaggio MQL4 conteneva solo tre funzioni predefinite che potevano essere utilizzate nell' indicatore, script o Expert Advisor (non tenendo conto del file include i file *.mqh e dei file della libreria). In MQL5 non esistono tali funzioni, ma ci sono loro analoghi. La tabella mostra la corrispondenza approssimativa delle funzioni.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

Le funzioni OnInit e OnDeinit svolgono lo stesso ruolo di init e deinit in MQL4 - sono progettati per individuare il codice, che deve essere eseguito durante l'inizializzazione e deinizializzazione dei programmi MQL5. Di conseguenza è possibile rinominare solo le funzioni, o lasciarle come sono, ma aggiungere chiamate di queste funzioni nei posti corrispondenti.

Esempio:

```
void OnInit()
  {
//--- Richiamo della funzione in fase di inizializzazione
```

```
   init();
  }
void OnDeinit(const int reason)
  {
//--- Chiamata di Funzione su deinizializzazione
   deinit();
//---
  }
```

La funzione start, è sostituita da OnStart solo negli script. Negli Expert Advisors ed Indicatori dovrebbe essere rinominata OnTick ed OnCalculate, rispettivamente. Il codice che deve essere eseguito durante un'operazione del programma mql5 dovrebbe essere collocato in queste tre funzioni:

mql5-program	main function
script	OnStart
indicatore	OnCalculate
Expert Advisor	OnTick

Se l'indicatore o il codice script non contengono la funzione principale, o il nome della funzione differisce da quella richiesta, la chiamata di questa funzione non viene eseguita. Significa, che se il codice sorgente di uno script non contiene OnStart, tale codice verrà compilato come un Expert Advisor.

Se un codice indicatore non contiene la funzione OnCalculate, la compilazione di tale indicatore è impossibile.

Variabili predefinite

In MQL5 non ci sono tali variabili predefinite come Ask, Bid, Bars. Le variabili Point e Digits hanno una nomenclatura leggermente diversa:

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

Accesso alle TimeSeries

In MQL5 non ci sono tali timeseries predefinite come Open[], High[], Low[], Close[], Volume[] e Time[]. La profondità necessaria di una TimeSeries può ora essere impostata tramite le corrispondenti funzioni per accedere alle timeseries.

Expert Advisors

gli Expert Advisors in MQL5 non richiedono la presenza obbligatoria di funzioni che gestiscono gli eventi di una ricezione di un nuovo tick - onTick, come era in MQL4 (la funzione di start in MQL4 viene eseguita quando un nuovo tick viene ricevuto). In MQL5 gli Expert Advisor possono contenere predefinite funzioni di gestione di diversi tipi di eventi:

- OnTick - ricezione di un nuovo tick;
- OnTimer - evento timer;
- OnTrade - evento trade;
- OnChartEvent - eventi di input dalla tastiera e mouse, eventi di un oggetto grafico in movimento, eventi di completamento di modifica del testo nel campo di immissione dell'oggetto LabelEdit;
- OnBookEvent - eventi di cambio di status del Depth of Market.

Custom Indicators

In MQL4, il numero di buffer indicatore è limitato e non può superare 8. In MQL5 ci sono limitazioni, ma va ricordato che ciascun buffer indicatore richiede l'allocazione di una certa parte di memoria per la sua posizione nel terminale, così la nuova possibilità non verrebbe abusata.

MQL4 offriva solo 6 tipi di plottaggio di indicatore personalizzato, mentre MQL5 offre ora 18 stili di disegno. I nomi dei tipi di disegno non sono cambiati, ma l'ideologia della rappresentazione grafica degli indicatori è cambiata significativamente.

La direzione di indicizzazione nei buffers indicatore anche, è diversa. Per impostazione predefinita, in MQL5 tutti i buffer indicatore hanno il comportamento di comuni array, cioè l' elemento indicizzato 0 è il più antico della cronistoria, e con l'aumento di indice, si passa dai dati più vecchi a quelli più recenti.

L'unica funzione per lavorare con gli indicatori personalizzati che è stata preservata dall' MQL4 è SetIndexBuffer. Ma la sua chiamata è cambiata, ora è necessario specificare il tipo di dati da memorizzare in un array, associato al buffer indicatore.

Le proprietà degli indicatori personalizzati anche si sono modificate ed ampliate. Nuove funzioni per accedere alle timeseries sono state aggiunte, quindi il totale algoritmo di calcolo deve essere riconsiderato.

Oggetti grafici

Il numero di oggetti grafici in MQL5 è stato notevolmente aumentato. Inoltre, gli oggetti grafici possono essere posizionati nel tempo con la precisione di un secondo in un grafico di qualsiasi timeframe - ora i punti di ancoraggio oggetto non sono arrotondati al tempo di barra di apertura nella tabella prezzi.

Per gli oggetti freccia, di Testo e Label ora è possibile specificare metodi di legatura, e per gli oggetti Label, Button, Chart, Bitmap Label ed Edit, è possibile impostare l' angolo del grafico a cui è collegato un oggetto.

List of MQL5 Functions

All MQL5 functions in alphabetical order.

Funzione	Azione	Section
AccountInfoDouble	Restituisce un valore di tipo double della corrispondente proprietà dell' account	Informazioni account
AccountInfoInteger	Restituisce un valore di tipo integer (bool, int o long) della corrispondente proprietà dell' account	Informazioni account
AccountInfoString	Restituisce un valore di tipo stringa della corrispondente proprietà dell'account	Informazioni account
acos	Restituisce l'arcocoseno di x in radianti	Funzioni Matematiche
Alert	Mostra un messaggio in una finestra separata.	Funzioni Comuni
ArrayBsearch	Searches for a specified value in a multidimensional numeric array sorted ascending	Funzioni di Array
ArrayCompare	Restituisce il risultato del confronto tra due array di tipi semplici o strutture personalizzate senza oggetti complessi	Funzioni di Array
ArrayCopy	Copia un array in un altro	Funzioni di Array
ArrayFill	Riempie un array con il valore specificato	Funzioni di Array
ArrayFree	Libera il buffer di ogni array dinamico ed imposta a 0 il valore della dimensione zero.	Funzioni di Array
ArrayGetAsSeries	Controlla la direzione di indicizzazione degli array	Funzioni di Array
ArrayInitialize	Imposta tutti gli elementi di un array numerico in un singolo valore	Funzioni di Array
ArrayIsDynamic	Controlla se un array è dinamico	Funzioni di Array
ArrayIsSeries	Controlla se un array è una serie temporale	Funzioni di Array

Funzione	Azione	Section
ArrayMaximum	Searches for the largest element in the first dimension of a multidimensional numeric array	Funzioni di Array
ArrayMinimum	Searches for the lowest element in the first dimension of a multidimensional numeric array	Funzioni di Array
ArrayRange	Restituisce il numero di elementi nella dimensione specifica dell'array	Funzioni di Array
ArrayResize	Imposta la nuova grandezza nella prima dimensione dell' array	Funzioni di Array
ArraySetAsSeries	Imposta la direzione di indicizzazione dell' array	Funzioni di Array
ArraySize	Restituisce il numero di elementi dell'array	Funzioni di Array
ArraySort	Ordinamento di array numerico per la prima dimensione	Funzioni di Array
ArrayPrint	Prints an array of a simple type or a simple structure into journal	Funzioni di Array
ArrayInsert	Inserisce il numero specificato di elementi da un array sorgente ad uno ricevente a partire da un indice specificato	Funzioni di Array
ArrayRemove	Rimuove il numero specificato di elementi dall'array iniziando con un indice specificato	Funzioni di Array
ArrayReverse	Inverte il numero specificato di elementi nell'array iniziando con un indice specificato	Funzioni di Array
ArraySwap	Scambia il contenuto di due array dinamici dello stesso tipo	Funzioni di Array
asin	Restituisce l'arcoseno di x in radianti	Funzioni Matematiche
atan	Restituisce l'arcotangente di x in radianti	Funzioni Matematiche
Bars	Restituisce il numero di barre dello storico di un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori

Funzione	Azione	Section
BarsCalculated	Restituisce il numero di dati calcolati in un buffer indicatore, oppure -1 in caso di errore (dati non sono stati ancora calcolati)	Accesso alle Timeseries ed Indicatori
CalendarCountryById	Ottiene una descrizione del Paese tramite il suo ID	Economic Calendar
CalendarEventById	Ottiene una descrizione dell'evento tramite il suo ID	Economic Calendar
CalendarValueById	Ottiene una descrizione del valore dell'evento tramite il suo ID	Economic Calendar
CalendarCountries	Ottiene la serie di nomi dei Paesi disponibili nel calendario	Economic Calendar
CalendarEventByCountry	Ottiene l'array delle descrizioni di tutti gli eventi disponibili nel calendario con un codice Paese specificato	Economic Calendar
CalendarEventByCurrency	Ottiene l'array di descrizioni di tutti gli eventi disponibili nel calendario in base ad una valuta specificata	Economic Calendar
CalendarValueHistoryByEvent	Ottiene l'array di valori per tutti gli eventi in un intervallo di tempo specificato da un ID evento	Economic Calendar
CalendarValueHistory	Ottiene l'array di valori per tutti gli eventi in un intervallo di tempo specificato con la possibilità di ordinare per Paese e/o valuta	Economic Calendar
CalendarValueLastByEvent	Ottiene l'array di valori di eventi in base al suo ID dallo stato del database del calendario con un change_id specificato	Economic Calendar
CalendarValueLast	Ottiene l'array di valori per tutti gli eventi con la possibilità di ordinare per Paese e/o valuta dallo stato del database del calendario con un change_id specificato	Economic Calendar
ceil	Restituisce il valore numerico integer(intero) più vicino dall'alto	Funzioni Matematiche

Funzione	Azione	Section
CharArrayToString	Conversione del codice simbolo (ansi) in un array costituito da un-simbolo	Funzioni di Conversione
ChartApplyTemplate	Applica un modello specifico da un file specificato al grafico	Operazioni col Grafico
ChartClose	Chiude il grafico specificato	Operazioni col Grafico
ChartFirst	Restituisce l'ID del primo grafico del terminale client	Operazioni col Grafico
ChartGetDouble	Restituisce il valore double della proprietà del grafico specificato	Operazioni col Grafico
ChartGetInteger	Restituisce il valore integer della proprietà del grafico specificato	Operazioni col Grafico
ChartGetString	Restituisce il valore string della proprietà del grafico specificato	Operazioni col Grafico
ChartID	Restituisce l'ID del grafico corrente	Operazioni col Grafico
ChartIndicatorAdd	Aggiunge un indicatore con l'handle specificato in una finestra del grafico specificato	Operazioni col Grafico
ChartIndicatorDelete	Rimuove un indicatore con un nome specificato dalla finestra del grafico specificato	Operazioni col Grafico
ChartIndicatorGet	Restituisce l'handle dell'indicatore con il nome breve specificato nella finestra del grafico specificato	Operazioni col Grafico
ChartIndicatorName	Restituisce il nome breve dell'indicatore per il numero nella lista degli indicatori nella finestra del grafico specificato	Operazioni col Grafico
ChartIndicatorsTotal	Restituisce il numero di tutti gli indicatori applicati alla finestra del grafico specificato.	Operazioni col Grafico
ChartNavigate	Esegue lo slittamento del grafico indicato, per il numero specificato di barre rispetto alla posizione specificata nel grafico	Operazioni col Grafico
ChartNext	Restituisce l'ID del grafico successivo a quello specificato	Operazioni col Grafico

Funzione	Azione	Section
ChartOpen	Apre un nuovo grafico con il simbolo e periodo specificato	Operazioni col Grafico
CharToString	Conversione di un codice di simboli in una stringa di un carattere	Funzioni di Conversione
ChartPeriod	Restituisce il valore del periodo del grafico specificato	Operazioni col Grafico
ChartPriceOnDropped	Restituisce la coordinata prezzo del punto nel grafico, per cui è stato allegato l'Expert Advisor o Script.	Operazioni col Grafico
ChartRedraw	Chiama un ridisegno forzato di un grafico specificato	Operazioni col Grafico
ChartSaveTemplate	Salva le impostazioni correnti grafico in un modello con un nome specificato	Operazioni col Grafico
ChartScreenShot	Fornisce uno screenshot del grafico del suo stato attuale in formato GIF, PNG o BMP a seconda estensione specificata	Operazioni col Grafico
ChartSetDouble	Imposta il valore double per una proprietà corrispondente del grafico specificato	Operazioni col Grafico
ChartSetInteger	Imposta il valore integer (datetime, int, color, bool o char) per una proprietà corrispondente del grafico specificato	Operazioni col Grafico
ChartSetString	Imposta il valore di stringa per una proprietà corrispondente del grafico specificato	Operazioni col Grafico
ChartSetSymbolPeriod	Cambia il valore di simbolo ed il periodo di grafico specificato	Operazioni col Grafico
ChartSymbol	Restituisce il nome del simbolo del grafico specificato	Operazioni col Grafico
ChartTimeOnDropped	Restituisce la coordinata temporaledel punto di grafico, per cui è stato allegato l'Expert Advisor o lo Script.	Operazioni col Grafico
ChartTimePriceToXY	Converte le coordinate di un grafico dalla rappresentazione	Operazioni col Grafico

Funzione	Azione	Section
	tempo/prezzo alle coordinate X e Y	
ChartWindowFind	Restituisce il numero di una sottofinestra in cui è disegnato un indicatore	Operazioni col Grafico
ChartWindowOnDropped	Restituisce il numero (indice) della sottofinestra grafico, per cui è stato allegato l'Expert Advisor o Script.	Operazioni col Grafico
ChartXOnDropped	Restituisce la coordinata X del punto di grafico, per cui è stato allegato l' Expert Advisor o lo Script.	Operazioni col Grafico
ChartXYToTimePrice	Converte le coordinate X e Y su un grafico dei valori di tempo e di prezzo	Operazioni col Grafico
ChartYOnDropped	Restituisce la coordinata Y del punto di grafico, per cui è stato allegato l' Expert Advisor o lo Script.	Operazioni col Grafico
CheckPointer	Restituisce il tipo di pontatore oggetto	Funzioni Comuni
CLBufferCreate	Crea un buffer di OpenCL	Lavorare con OpenCL
CLBufferFree	Elimina un buffer OpenCL	Lavorare con OpenCL
CLBufferRead	Legge un buffer OpenCL in un array	Lavorare con OpenCL
CLBufferWrite	Scrive un array in un buffer OpenCL	Lavorare con OpenCL
CLContextCreate	Crea un contesto OpenCL	Lavorare con OpenCL
CLContextFree	Rimuove un contesto OpenCL	Lavorare con OpenCL
CLExecute	Esegue un programma OpenCL	Lavorare con OpenCL
CLGetDeviceInfo	Riceve la proprietà del dispositivo dal driver di OpenCL	Lavorare con OpenCL
CLGetInfoInteger	Restituisce il valore di una proprietà integer per un oggetto OpenCL o dispositivo	Lavorare con OpenCL
CLHandleType	Restituisce il tipo di un handle OpenCL come valore	Lavorare con OpenCL

Funzione	Azione	Section
	dell'enumerazione ENUM_OPENCL_HANDLE_TYPE	
CLKernelCreate	Crea una funzione di avvio OpenCL	Lavorare con OpenCL
CLKernelFree	Rimuove una funzione di avvio OpenCL	Lavorare con OpenCL
CLProgramCreate	Crea un programma di OpenCL da un codice sorgente	Lavorare con OpenCL
CLProgramFree	Rimuove un programma di OpenCL	Lavorare con OpenCL
CLSetKernelArg	Imposta un parametro per la funzione OpenCL	Lavorare con OpenCL
CLSetKernelArgMem	Imposta un buffer OpenCL come parametro della funzione OpenCL	Lavorare con OpenCL
ColorToARGB	Conversione del tipo colore in tipo uint per ricevere la rappresentazione ARGB del colore.	Funzioni di Conversione
ColorToString	Conversione del valore del colore in stringa come "R,G,B"	Funzioni di Conversione
Comment	Manda in output un commento, nell'angolo superiore sinistro del grafico	Funzioni Comuni
CopyBuffer	Mette in un array i dati di un buffer specificato da un indicatore specificato	Accesso alle Timeseries ed Indicatori
CopyClose	Mette in un array i dati storici sul prezzo di chiusura della barra per un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori
CopyHigh	Mette in un array i dati storici sul prezzo massimo della barra per un periodo e simbolo specificati	Accesso alle Timeseries ed Indicatori
CopyLow	Mette in un array i dati storici sul prezzo minimo della barra per un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori
CopyOpen	Mette in un array i dati storici sul prezzo di apertura della barra per un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori
CopyRates	Mette in un array i dati storici della struttura Rates per un	Accesso alle Timeseries ed Indicatori

Funzione	Azione	Section
	simbolo e periodo specificati	
CopyRealVolume	Mette in un array i dati storici sui volumi trade per un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori
CopySpread	Mette in un array i dati storici sugli spread per un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori
CopyTicks	Gets ticks accumulated by the terminal for the current working session into an array	Accesso alle Timeseries ed Indicatori
CopyTickVolume	Mette in un array i dati storici relativi ai volumi tick per un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori
CopyTime	Mette in un array i dati storici dell'orario di apertura della barra per un simbolo e periodo specificati	Accesso alle Timeseries ed Indicatori
cos	Restituisce il coseno di un numero	Funzioni Matematiche
CryptDecode	Performs the inverse transformation of the data from array	Funzioni Comuni
CryptEncode	Transforms the data from array with the specified method	Funzioni Comuni
CustomSymbolCreate	Creare un simbolo personalizzato con il nome specificato nel gruppo specificato	Simboli Personalizzati
CustomSymbolDelete	Eliminare un simbolo personalizzato con il nome specificato	Simboli Personalizzati
CustomSymbolSetInteger	Impostare il valore di proprietà del tipo intero per un simbolo personalizzato	Simboli Personalizzati
CustomSymbolSetDouble	Impostare il valore di proprietà del tipo reale per un simbolo personalizzato	Simboli Personalizzati
CustomSymbolSetString	Impostare il valore della proprietà di tipo stringa per un simbolo personalizzato	Simboli Personalizzati
CustomSymbolSetMarginRate	Impostare i tassi di margine in base al tipo di ordine e alla	Simboli Personalizzati

Funzione	Azione	Section
	direzione di un simbolo personalizzato	
CustomSymbolSetSessionQuote	Imposta l'ora di inizio e fine della sessione di quotazioni specificata per il simbolo ed il giorno della settimana specificati	Simboli Personalizzati
CustomSymbolSetSessionTrade	Impostare l'ora di inizio e fine della sessione di trading specificata per il simbolo e il giorno della settimana specificati	Simboli Personalizzati
CustomRatesDelete	Elimina tutte le barre dalla cronologia dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato	Simboli Personalizzati
CustomRatesReplace	Sostituisce completamente lo storico dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato con i dati dell'array tipo MqlRates	Simboli Personalizzati
CustomRatesUpdate	Aggiunge barre mancanti allo storico dei simboli custom e sostituisce i dati esistenti con quelli dell'array tipo MqlRates	Simboli Personalizzati
CustomTicksAdd	Aggiunge i dati da un array del tipo MqlTick alla cronologia dei prezzi di un simbolo personalizzato. Il simbolo personalizzato deve essere selezionato nella finestra del Watch Market.	Simboli Personalizzati
CustomTicksDelete	Eliminare tutti i ticks dallo storico dei prezzi del simbolo personalizzato nell'intervallo di tempo specificato	Simboli Personalizzati
CustomTicksReplace	Sostituisce completamente lo storico dei prezzi del simbolo personalizzato entro l'intervallo di tempo specificato con i dati dell'array di tipo MqlTick	Simboli Personalizzati
CustomBookAdd	Passa lo status del Depth of Market per un simbolo personalizzato	Simboli Personalizzati

Funzione	Azione	Section
DatabaseOpen	Apre o crea un database in un file specificato	Lavorare con i database
DatabaseClose	Chiude un database	Lavorare con i database
DatabaseImport	Importa i dati da un file in una tabella	Lavorare con i database
DatabaseExport	Esporta una tabella o un risultato di esecuzione di una richiesta SQL in un file CSV	Lavorare con i database
DatabasePrint	Stampa una tabella o un risultato di esecuzione di una richiesta SQL nel journal degli experts	Lavorare con i database
DatabaseTableExists	Verifica la presenza della tabella in un database	Lavorare con i database
DatabaseExecute	Esegue una richiesta ad un database specificato	Lavorare con i database
DatabasePrepare	Crea un handle di una richiesta, che può quindi essere eseguita utilizzando DatabaseRead()	Lavorare con i database
DatabaseReset	Reimposta una richiesta, come dopo aver chiamato DatabasePrepare()	Lavorare con i database
DatabaseBind	Imposta un valore di parametro in una richiesta	Lavorare con i database
DatabaseBindArray	Imposta un array come valore di parametro	Lavorare con i database
DatabaseRead	Passa alla voce successiva a seguito di una richiesta	Lavorare con i database
DatabaseFinalize	Rimuove una richiesta creata in DatabasePrepare()	Lavorare con i database
DatabaseTransactionBegin	Inizia l'esecuzione della transazione	Lavorare con i database
DatabaseTransactionCommit	Completa l'esecuzione della transazione	Lavorare con i database
DatabaseTransactionRollback	Ripristina le transazioni	Lavorare con i database
DatabaseColumnsCount	Ottiene il numero di campi in una richiesta	Lavorare con i database
DatabaseColumnName	Ottiene un nome campo per indice	Lavorare con i database

Funzione	Azione	Section
DatabaseColumnType	Ottiene un tipo di campo per indice	Lavorare con i database
DatabaseColumnSize	Ottiene una dimensione del campo in byte	Lavorare con i database
DatabaseColumnText	Ottiene un valore di campo come stringa dal record corrente	Lavorare con i database
DatabaseColumnInteger	Ottiene il valore del tipo int dal record corrente	Lavorare con i database
DatabaseColumnLong	Ottiene il valore di tipo long dal record corrente	Lavorare con i database
DatabaseColumnDouble	Ottiene il valore di tipo double dal record corrente	Lavorare con i database
DatabaseColumnBlob	Ottiene un valore di campo come un array dal record corrente	Lavorare con i database
DebugBreak	Breakpoint del programma nel debugging	Funzioni Comuni
Digits	Restituisce il numero di cifre decimali determinando l'accuratezza del valore prezzo del corrente simbolo del grafico	Verifica Stato
DoubleToString	Conversione di un valore numerico ad una riga di testo con una precisione specificata	Funzioni di Conversione
DXContextCreate	Crea un contesto grafico per il rendering di frame di una grandezza specificata	Lavorare con DirectX
DXContextSetSize	Modifica le dimensioni di un frame di un contesto grafico creato in DXContextCreate()	Lavorare con DirectX
DXContextSetSize	Ottiene una dimensione del frame di un contesto grafico creato in DXContextCreate()	Lavorare con DirectX
DXContextClearColors	Imposta un colore specificato su tutti i pixel per il buffer di rendering	Lavorare con DirectX
DXContextClearDepth	Cancella il buffer di profondità	Lavorare con DirectX
DXContextGetColors	Ottiene un'immagine di dimensioni specificate ed offset da un contesto grafico	Lavorare con DirectX

Funzione	Azione	Section
DXContextGetDepth	Ottiene il buffer di profondità di un fotogramma renderizzato	Lavorare con DirectX
DXBufferCreate	Crea un buffer di un tipo specificato basato su un array di dati	Lavorare con DirectX
DXTextureCreate	Crea una trama 2D da un rettangolo di una dimensione specificata tagliato da un'immagine passata	Lavorare con DirectX
DXInputCreate	Crea input shader	Lavorare con DirectX
DXInputSet	Imposta gli input dello shader	Lavorare con DirectX
DXShaderCreate	Crea uno shader di un tipo specificato	Lavorare con DirectX
DXShaderSetLayout	Imposta il layout dei vertici per il vertex shader	Lavorare con DirectX
DXShaderInputsSet	Imposta gli input dello shader	Lavorare con DirectX
DXShaderTexturesSet	Imposta le texture dello shader	Lavorare con DirectX
DXDraw	Esegue il rendering dei vertici del vertex buffer impostato in DXBufferSet()	Lavorare con DirectX
DXDrawIndexed	Esegue il rendering delle primitive grafiche descritte dall' index buffer da DXBufferSet()	Lavorare con DirectX
DXPrimiveTopologySet	Imposta il tipo di primitive per il rendering utilizzando DXDrawIndexed()	Lavorare con DirectX
DXBufferSet	Imposta un buffer per il rendering corrente	Lavorare con DirectX
DXShaderSet	Imposta uno shader per il rendering	Lavorare con DirectX
DXHandleType	Restituisce un tipo di handle	Lavorare con DirectX
DXRelease	Rilascia un handle	Lavorare con DirectX
EnumToString	Conversione di un valore di enumerazione di qualsiasi tipo a stringa	Funzioni di Conversione
EventChartCustom	Genera un evento personalizzato per il grafico specificato	Funzioni Eventi

Funzione	Azione	Section
EventKillTimer	Interrompe la generazione di eventi dal timer nel grafico corrente	Funzioni Eventi
EventSetMillisecondTimer	Avvia il generatore evento del timer ad alta risoluzione con un periodo di meno di 1 secondo per il chart corrente	Funzioni Eventi
EventSetTimer	Avvia il generatore di eventi timer con la periodicità specificata per il grafico corrente	Funzioni Eventi
exp	Restituisce l'esponente di un numero	Funzioni Matematiche
ExpertRemove	Ferma l' Expert Advisor e lo decarica dal grafico	Funzioni Comuni
fabs	Restituisce il valore assoluto (modulo) del valore numerico specificato	Funzioni Matematiche
FileClose	Chiude un file aperto in precedenza	Funzioni con i File
FileCopy	Copia il file originale da una cartella locale o condivisa in un altro file	Funzioni con i File
FileDelete	Elimina un file specificato	Funzioni con i File
FileFindClose	Chiude l'handle di ricerca	Funzioni con i File
FileFindFirst	Avvia la ricerca di file in una directory secondo il filtro specificato	Funzioni con i File
FileFindNext	Continua la ricerca iniziata dalla funzione FileFindFirst()	Funzioni con i File
FileFlush	Scrive su un disco tutti i dati rimasti nel buffer input/output del file	Funzioni con i File
FileGetInteger	Ottiene una proprietà integer di un file	Funzioni con i File
FileIsEnding	Definisce la fine di un file nel processo di lettura	Funzioni con i File
FileIsExist	Controlla l'esistenza di un file	Funzioni con i File
FileIsLineEndin	Definisce la fine di una riga in un file di testo nel processo di lettura	Funzioni con i File

Funzione	Azione	Section
FileMove	Sposta o rinomina un file	Funzioni con i File
FileOpen	Apre un file con nome e flag specificati	Funzioni con i File
FileReadArray	Legge array di qualsiasi tipo ad eccezione di string da file di tipo BIN	Funzioni con i File
FileReadBool	Legge dal file di tipo CSV una stringa a partire dalla posizione corrente fino ad un delimitatore (o fino alla fine di una riga di testo), e converte la stringa di lettura per un valore di tipo bool	Funzioni con i File
FileReadDatetime	Legge dal file di tipo CSV una stringa in uno dei formati: "AAAA.MM.GG. HH:MM:SS", "AAAA.MM.GG." o "HH:MM:SS" - e la converte in un valore datetime	Funzioni con i File
FileReadDouble	Legge un valore double dalla posizione corrente del puntatore del file	Funzioni con i File
FileReadFloat	Legge un valore float dalla posizione corrente del puntatore del file	Funzioni con i File
FileReadInteger	Legge valori int, short o char, dalla posizione corrente del puntatore del file	Funzioni con i File
FileReadLong	Legge un valore di tipo long dalla posizione corrente del puntatore del file	Funzioni con i File
FileReadNumber	Legge dal file di tipo CSV una stringa a partire dalla posizione corrente fino ad un delimitatore (o fino alla fine di una riga di testo), e converte la stringa di lettura in valore double	Funzioni con i File
FileReadString	Legge una stringa dalla posizione corrente di un puntatore ad un file da un file	Funzioni con i File
FileReadStruct	Legge il contenuto di un file binario in una struttura passato come parametro, dalla posizione corrente del puntatore del file	Funzioni con i File

Funzione	Azione	Section
FileSeek	Sposta la posizione del puntatore del file di un determinato numero di byte rispetto alla posizione specificata	Funzioni con i File
FileSize	Restituisce la grandezza del corrispondente file aperto	Funzioni con i File
FileTell	Restituisce la posizione corrente del puntatore del file del corrispondente file aperto	Funzioni con i File
FileWrite	Scrive dati in un file di tipo CSV o TXT	Funzioni con i File
FileWriteArray	Scrive array di qualsiasi tipo ad eccezione di string in un file di tipo BIN	Funzioni con i File
FileWriteDouble	Scrive il valore di tipo double dalla posizione corrente di un puntatore di un file in un file binario	Funzioni con i File
FileWriteFloat	Scrive il valore di tipo float dalla posizione corrente di un puntatore di un file in un file binario	Funzioni con i File
FileWriteInteger	Scrive il valore di tipo int dalla posizione corrente di un puntatore di un file in un file binario	Funzioni con i File
FileWriteLong	Scrive il valore di tipo long dalla posizione corrente di un puntatore di un file in un file binario	Funzioni con i File
FileWriteString	Scrive il valore di un parametro string in un file BIN o TXT a partire dalla posizione corrente del puntatore del file	Funzioni con i File
FileWriteStruct	Scrive il contenuto di una struttura passata come parametro in un file binario, a partire dalla posizione corrente del puntatore del file	Funzioni con i File
floor	Restituisce il valore numerico integer(intero) più vicino dal basso	Funzioni Matematiche
fmax	Restituisce il valore massimo dei due valori numerici	Funzioni Matematiche

Funzione	Azione	Section
fmin	Restituisce il valore minimo dei due valori numerici	Funzioni Matematiche
fmod	Restituisce il resto vero dopo la divisione di due numeri	Funzioni Matematiche
FolderClean	Elimina tutti i file presenti nella cartella specificata	Funzioni con i File
FolderCreate	Crea una cartella nella directory dei file	Funzioni con i File
FolderDelete	Rimuove una directory selezionata. Se la cartella non è vuota, allora non può essere rimossa	Funzioni con i File
FrameAdd	Aggiunge un frame con dati	Lavorare con i risultati di ottimizzazione
FrameFilter	Imposta il frame di lettura filtro e sposta il puntatore all'inizio	Lavorare con i risultati di ottimizzazione
FrameFirst	Sposta un puntatore di lettura frame all'inizio e resetta il filtro impostato precedentemente	Lavorare con i risultati di ottimizzazione
FrameInputs	Riceve i parametri di input, in cui il frame viene formato	Lavorare con i risultati di ottimizzazione
FrameNext	Legge un frame e sposta il puntatore al successivo	Lavorare con i risultati di ottimizzazione
GetLastError	Restituisce l'ultimo errore	Verifica Stato
GetPointer	Restituisce il pointer dell'oggetto	Funzioni Comuni
GetTickCount	Restituisce il numero di millisecondi che sono passati dal momento in cui il sistema è partito	Funzioni Comuni
GlobalVariableCheck	Controlla l'esistenza di una variabile globale con il nome specificato	Variabili Globali del Terminale
GlobalVariableDel	Elimina una variabile globale	Variabili Globali del Terminale
GlobalVariableGet	Restituisce il valore di una variabile globale	Variabili Globali del Terminale
GlobalVariableName	Restituisce il nome di una variabile globale dal suo numero	Variabili Globali del Terminale

Funzione	Azione	Section
	ordinale nell'elenco delle variabili globali	
GlobalVariablesDeleteAll	Elimina le variabili globali con il prefisso specificato nel loro nome	Variabili Globali del Terminale
GlobalVariableSet	Imposta il nuovo valore in una variabile globale	Variabili Globali del Terminale
GlobalVariableSetOnCondition	Imposta il nuovo valore dell'esistente variabile globale per condizione	Variabili Globali del Terminale
GlobalVariablesFlush	Salva forzatamente contenuto di tutte le variabili globali nel disco	Variabili Globali del Terminale
GlobalVariablesTotal	Restituisce il numero totale di variabili globali	Variabili Globali del Terminale
GlobalVariableTemp	Imposta il nuovo valore in una variabile globale, che esiste solo nella sessione corrente del terminale	Variabili Globali del Terminale
GlobalVariableTime	Restituisce l'orario dell'ultimo accesso alla variabile globale	Variabili Globali del Terminale
HistoryDealGetDouble	Restituisce la proprietà richiesta di un addare nella cronistoria (double)	Funzioni di Trade
HistoryDealGetInteger	Restituisce la proprietà richiesta di un affare nella cronistoria (datetime o int)	Funzioni di Trade
HistoryDealGetString	Restituisce la proprietà richiesta di un addare nella cronistoria (string)	Funzioni di Trade
HistoryDealGetTicket	Restituisce un ticket di un affare corrispondente nella cronistoria	Funzioni di Trade
HistoryDealSelect	Seleziona un affare nella cronistoria per l'ulteriore chiama attraverso funzioni appropriate	Funzioni di Trade
HistoryDealsTotal	Restituisce il numero degli affari nella cronistoria	Funzioni di Trade
HistoryOrderGetDouble	Restituisce la proprietà richiesta di un ordine nella cronistoria (doppia)	Funzioni di Trade

Funzione	Azione	Section
HistoryOrderGetInteger	Restituisce la proprietà richiesta di un ordine nella cronistoria (datetime o int)	Funzioni di Trade
HistoryOrderGetString	Restituisce la proprietà richiesta di un ordine nella cronistoria (stringa)	Funzioni di Trade
HistoryOrderGetTicket	Restituisce il ticket dell'ordine corrispondente all' ordine nella cronistoria	Funzioni di Trade
HistoryOrderSelect	Seleziona un ordine nella cronistoria per un ulteriore lavoro con esso	Funzioni di Trade
HistoryOrdersTotal	Restituisce il numero di ordini nella cronistoria	Funzioni di Trade
HistorySelect	Recupera la cronistoria delle transazioni e degli ordini per il periodo di tempo specificato del server time	Funzioni di Trade
HistorySelectByPosition	Richiede la cronistoria delle offerte con un determinato position identifier.	Funzioni di Trade
iBars	Restituisce il numero di barre di un simbolo e di un periodo corrispondenti, disponibili nello storico	Accesso alle Timeseries ed Indicatori
iBarShift	Restituisce l'indice della barra corrispondente al tempo/orario specificato	Accesso alle Timeseries ed Indicatori
iClose	Restituisce il prezzo Close (di chiusura) della barra (indicato dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iHigh	Restituisce il prezzo High della barra (indicata dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iHighest	Restituisce l'indice del valore più alto trovato sul chart corrispondente (spostamento relativo alla barra corrente)	Accesso alle Timeseries ed Indicatori
iLow	Restituisce il prezzo Low della barra (indicato dal parametro	Accesso alle Timeseries ed Indicatori

Funzione	Azione	Section
	'shift') sul chart corrispondente	
iLowest	Restituisce l'indice del valore più piccolo trovato sul chart corrispondente (slittamento relativo alla barra corrente)	Accesso alle Timeseries ed Indicatori
iOpen	Restituisce il prezzo Open della barra (indicata dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iTime	Restituisce il tempo di apertura della barra (indicato dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iTickVolume	Restituisce il volume tick della barra (indicato dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iRealVolume	Restituisce il volume reale della barra (indicato dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iVolume	Restituisce il volume tick della barra (indicato dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iSpread	Restituisce il valore di spread della barra (indicato dal parametro 'shift') sul chart corrispondente	Accesso alle Timeseries ed Indicatori
iAD	Accumulation/Distribution	Indicatori Tecnici
iADX	Average Directional Index	Indicatori Tecnici
iADXWilder	Average Directional Index by Welles Wilder	Indicatori Tecnici
iAlligator	Alligator	Indicatori Tecnici
iAMA	Adaptive Moving Average	Indicatori Tecnici
iAO	Awesome Oscillator	Indicatori Tecnici
iATR	Average True Range	Indicatori Tecnici
iBands	Bollinger Bands®	Indicatori Tecnici
iBearsPower	Bears Power	Indicatori Tecnici
iBullsPower	Bulls Power	Indicatori Tecnici
iBWMFI	Market Facilitation Index by Bill Williams	Indicatori Tecnici

Funzione	Azione	Section
iCCI	Commodity Channel Index	Indicatori Tecnici
iChaikin	Chaikin Oscillator	Indicatori Tecnici
iCustom	Custom indicator	Indicatori Tecnici
iDEMA	Double Exponential Moving Average	Indicatori Tecnici
iDeMarker	DeMarker	Indicatori Tecnici
iEnvelopes	Envelopes	Indicatori Tecnici
iForce	Force Index	Indicatori Tecnici
iFractals	Fractals	Indicatori Tecnici
iFrAMA	Fractal Adaptive Moving Average	Indicatori Tecnici
iGator	Gator Oscillator	Indicatori Tecnici
iIchimoku	Ichimoku Kinko Hyo	Indicatori Tecnici
iMA	Moving Average	Indicatori Tecnici
iMACD	Moving Averages Convergence-Divergence	Indicatori Tecnici
iMFI	Money Flow Index	Indicatori Tecnici
iMomentum	Momentum	Indicatori Tecnici
IndicatorCreate	Restituisce l'handle per l'indicatore tecnico specificato creato da una serie di parametri di tipo MqlParam	Accesso alle Timeseries ed Indicatori
IndicatorParameters	Basato sull' handler specificato, restituisce il numero di parametri di input dell'indicatore, nonché i valori e tipi dei parametri	Accesso alle Timeseries ed Indicatori
IndicatorRelease	Rimuove un handle indicatore e rilascia il blocco di calcolo dell'indicatore, se non è usato da nessun altro	Accesso alle Timeseries ed Indicatori
IndicatorSetDouble	Imposta il valore della proprietà dell' indicatore di tipo double	Indicatori Personalizzati
IndicatorSetInteger	Imposta il valore di una proprietà dell'indicatore di tipo int	Indicatori Personalizzati
IndicatorSetString	Imposta il valore della proprietà dell' indicatore di tipo string	Indicatori Personalizzati

Funzione	Azione	Section
IntegerToString	Conversione di int in una stringa di lunghezza predefinita	Funzioni di Conversione
iOBV	On Balance Volume	Indicatori Tecnici
iOsMA	Moving Average of Oscillator (MACD histogram)	Indicatori Tecnici
iRSI	Relative Strength Index	Indicatori Tecnici
iRVI	Relative Vigor Index	Indicatori Tecnici
iSAR	Parabolic Stop And Reverse System	Indicatori Tecnici
IsStopped	Restituisce true, se un programma mql5 è stato comandato di fermarsi dal suo funzionamento.	Verifica Stato
iStdDev	Standard Deviation	Indicatori Tecnici
iStochastic	Stochastic Oscillator	Indicatori Tecnici
iTEMA	Triple Exponential Moving Average	Indicatori Tecnici
iTriX	Triple Exponential Moving Averages Oscillator	Indicatori Tecnici
iVIDyA	Variable Index Dynamic Average	Indicatori Tecnici
iVolumes	Volumes	Indicatori Tecnici
iWPR	Williams' Percent Range	Indicatori Tecnici
log	Restituisce il logaritmo naturale	Funzioni Matematiche
log10	Restituisce il logaritmo di un numero da base 10	Funzioni Matematiche
MarketBookAdd	Fornisce l'apertura della Profondità di Mercato per un simbolo selezionato, e sottoscrive per ricevere le notifiche dei cambiamenti DOM	Market Info
MarketBookGet	Restituisce un array di strutture MqlBookInfo contenente records della Profondità di Mercato di un simbolo specifico	Market Info
MarketBookRelease	Fornisce la chiusura della Profondità di Mercato per un simbolo selezionato, ed annulla la	Market Info

Funzione	Azione	Section
	sottoscrizione per ricevere le notifiche dei cambiamenti DOM	
MathAbs	Restituisce il valore assoluto (modulo) del valore numerico specificato	Funzioni Matematiche
MathArccos	Restituisce l'arcocoseno di x in radianti	Funzioni Matematiche
MathArcsin	Restituisce l'arcoseno di x in radianti	Funzioni Matematiche
MathArctan	Restituisce l'arcotangente di x in radianti	Funzioni Matematiche
MathCeil	Restituisce il valore numerico integer(intero) più vicino dall'alto	Funzioni Matematiche
MathCos	Restituisce il coseno di un numero	Funzioni Matematiche
MathExp	Restituisce l'esponente di un numero	Funzioni Matematiche
MathFloor	Restituisce il valore numerico integer(intero) più vicino dal basso	Funzioni Matematiche
MathIsValidNumber	Verifica la correttezza di un numero reale	Funzioni Matematiche
MathLog	Restituisce il logaritmo naturale	Funzioni Matematiche
MathLog10	Restituisce il logaritmo di un numero da base 10	Funzioni Matematiche
MathMax	Restituisce il valore massimo dei due valori numerici	Funzioni Matematiche
MathMin	Restituisce il valore minimo dei due valori numerici	Funzioni Matematiche
MathMod	Restituisce il resto vero dopo la divisione di due numeri	Funzioni Matematiche
MathPow	Aumenta la base alla potenza specificata	Funzioni Matematiche
MathRand	Restituisce un valore pseudocasuale nell'intervallo da 0 a 32767	Funzioni Matematiche
MathRound	Arrotonda un valore al numero intero più vicino	Funzioni Matematiche
MathSin	Restituisce il seno di un numero	Funzioni Matematiche

Funzione	Azione	Section
MathSqrt	Restituisce una radice quadrata	Funzioni Matematiche
MathSrand	Consente di impostare il punto di partenza per la generazione di una serie di interi pseudocasuali	Funzioni Matematiche
MathTan	Restituisce la tangente di un numero	Funzioni Matematiche
MessageBox	Crea e mostra un box messaggi e lo gestisce	Funzioni Comuni
MQLInfoInteger	Restituisce il valore integer della corrispondente proprietà del programma mql5 che sta girando	Verifica Stato
MQLInfoString	Restituisce il valore stringa della corrispondente proprietà del programma mql5 che sta girando	Verifica Stato
MT5Initialize	Stabilisce una connessione con il terminale MetaTrader 5	MetaTrader per Python
MT5Shutdown	Chiude la connessione precedentemente stabilita al terminale MetaTrader 5	MetaTrader per Python
MT5TerminalInfo	Ottiene lo stato ed i parametri del terminale MetaTrader 5 collegato	MetaTrader per Python
MT5Version	Restituisce la versione del terminale MetaTrader 5	MetaTrader per Python
MT5CopyRatesFrom	Ottiene le barre dal terminale MetaTrader 5 a partire dalla data specificata	MetaTrader per Python
MT5CopyRatesFromPos	Ottiene le barre dal terminale MetaTrader 5 a partire dall'indice specificato	MetaTrader per Python
MT5CopyRatesRange	Ottiene le barre nell'intervallo di date specificato dal terminale MetaTrader 5	MetaTrader per Python
MT5CopyTicksFrom	Ottiene i tick tick dal terminale MetaTrader 5 a partire dalla data specificata	MetaTrader per Python
MT5CopyTicksRange	Ottiene i tick per l'intervallo di date specificato dal terminale MetaTrader 5	MetaTrader per Python

Funzione	Azione	Section
NormalizeDouble	Arrotondamento di un numero in virgola mobile a una precisione specificata	Funzioni di Conversione
ObjectCreate	Crea un oggetto del tipo specificato in una tabella specificata	Funzioni Oggetto
ObjectDelete	Rimuove l'oggetto con il nome specificato dalla tabella specificata (dalla sottofinestra grafico specificata)	Funzioni Oggetto
ObjectFind	Cerca un oggetto con l'ID specificato, per nome	Funzioni Oggetto
ObjectGetDouble	Restituisce il doppio valore della proprietà dell'oggetto corrispondente	Funzioni Oggetto
ObjectGetInteger	Restituisce il valore intero della proprietà dell'oggetto corrispondente	Funzioni Oggetto
ObjectGetString	Restituisce il valore stringa della proprietà dell'oggetto corrispondente	Funzioni Oggetto
ObjectGetTimeByValue	Restituisce il valore di tempo per il valore dell'oggetto prezzo specificato	Funzioni Oggetto
ObjectGetValueByTime	Restituisce il valore del prezzo di un oggetto per il periodo di tempo specificato	Funzioni Oggetto
ObjectMove	Cambia le coordinate del punto di ancoraggio dell' oggetto specificato	Funzioni Oggetto
ObjectName	Restituisce il nome di un oggetto del tipo corrispondente nella tabella specificata (grafico sottofinestra specificato)	Funzioni Oggetto
ObjectsDeleteAll	Rimuove tutti gli oggetti del tipo specificato dalla tabella specificata (dalla sottofinestra grafico specificata)	Funzioni Oggetto
ObjectSetDouble	Imposta il valore della proprietà dell'oggetto corrispondente	Funzioni Oggetto

Funzione	Azione	Section
ObjectSetInteger	Imposta il valore della proprietà dell'oggetto corrispondente	Funzioni Oggetto
ObjectSetString	Imposta il valore della proprietà dell'oggetto corrispondente	Funzioni Oggetto
ObjectsTotal	Restituisce il numero di oggetti del tipo specificato nella tabella specificata (grafico sottofinestra specificato)	Funzioni Oggetto
OnStart	La funzione è chiamata quando si verifica l'evento Start per eseguire azioni impostate nello script	Gestione degli Eventi
OnInit	La funzione è chiamata negli indicatori ed EA quando si verifica l'evento Init per inizializzare un programma MQL5 avviato	Gestione degli Eventi
OnDeinit	La funzione è chiamata negli indicatori ed EA quando si verifica l'evento Deinit per de-inizializzare un programma MQL5 avviato	Gestione degli Eventi
OnTick	La funzione è chiamata nell' EA quando si verifica l'evento NewTick per gestire una nuova quotazione	Gestione degli Eventi
OnCalculate	La funzione è chiamata negli indicatori quando si verifica l'evento Calculate per gestire i cambiamenti dei dati di prezzo	Gestione degli Eventi
OnTimer	La funzione è chiamata negli indicatori ed EA durante l'evento periodico Timer generato dal terminale a intervalli di tempo fissi	Gestione degli Eventi
OnTrade	La funzione è chiamata nell' EA durante l'evento generato Trade al termine di un'operazione di trading su un trade server	Gestione degli Eventi
OnTradeTransaction	La funzione è chiamata nell' EA quando si verifica l'evento TradeTransaction per elaborare un risultato di esecuzione di una richiesta di trade	Gestione degli Eventi

Funzione	Azione	Section
OnBookEvent	La funzione è chiamata nell'EA quando si verifica l'evento BookEvent per elaborare i cambiamenti nel market depth	Gestione degli Eventi
OnChartEvent	La funzione è chiamata in indicatori ed EA quando si verifica l'evento ChartEvent per elaborare le modifiche del grafico-chart effettuate da un utente o un programma MQL5	Gestione degli Eventi
OnTester	La funzione è chiamata nell'EA quando si verifica l'evento Tester per eseguire le azioni necessarie dopo aver testato un EA sui dati della cronistoria	Gestione degli Eventi
OnTesterInit	La funzione è chiamata nell' EA quando si verifica l'evento TesterInit per eseguire le azioni necessarie prima dell'ottimizzazione nel tester di strategia	Gestione degli Eventi
OnTesterDeinit	La funzione è chiamata nell'EA quando si verifica l'evento TesterDeinit dopo l'ottimizzazione EA nel tester di strategia	Gestione degli Eventi
OnTesterPass	La funzione è chiamata nell' EA quando si verifica l'evento TesterPass per gestire l'arrivo di un nuovo frame di dati, durante l'ottimizzazione EA nel tester di strategia	Gestione degli Eventi
OrderCalcMargin	Calcola il margine richiesto per il tipo di ordine specificato, nella valuta di deposito	Funzioni di Trade
OrderCalcProfit	Calcola il profitto in base ai parametri passati, nella valuta di deposito	Funzioni di Trade
OrderCheck	Verifica se ci sono fondi sufficienti per eseguire l' operazione di traderichiesta.	Funzioni di Trade
OrderGetDouble	Restituisce la proprietà richiesta dell'ordine (double)	Funzioni di Trade

Funzione	Azione	Section
OrderGetInteger	Restituisce la proprietà richiesta dell'ordine (datetime o int)	Funzioni di Trade
OrderGetString	Restituisce la proprietà richiesta dell'ordine (string)	Funzioni di Trade
OrderGetTicket	Restituisce il ticket di un ordine corrispondente	Funzioni di Trade
OrderSelect	Seleziona un ordine per un'ulteriore lavoro con esso	Funzioni di Trade
OrderSend	Invia richieste di trade ad un server	Funzioni di Trade
OrderSendAsync	Invia in modo asincrono richieste di trade senza attendere la risposta del trade, dal trade server	Funzioni di Trade
OrdersTotal	Restituisce il numero di ordini	Funzioni di Trade
ParameterGetRange	Riceve i dati sulla gamma di valori ed lo step del cambiamento per una variabile di input quando si ottimizza un Expert Advisor nello Strategy Tester	Lavorare con i risultati di ottimizzazione
ParameterSetRange	Specifica l'uso della variabile di input quando si ottimizza un Expert Advisor nello Strategy Tester: valore, step del cambiamento, i valori iniziali e finali	Lavorare con i risultati di ottimizzazione
Period	Restituisce il timeframe del corrente grafico	Verifica Stato
PeriodSeconds	Restituisce il numero di secondi nel periodo	Funzioni Comuni
PlaySound	Riproduce un file sonoro	Funzioni Comuni
PlotIndexGetInteger	Restituisce il valore della proprietà della linea dell'indicatore di tipo integer	Indicatori Personalizzati
PlotIndexSetDouble	Imposta il valore della proprietà dell' indicatore di tipo double	Indicatori Personalizzati
PlotIndexSetInteger	Imposta il valore di una proprietà dell'indicatore di tipo int	Indicatori Personalizzati

Funzione	Azione	Section
PlotIndexSetString	Imposta il valore di una proprietà dell'indicatore di tipo string	Indicatori Personalizzati
Point	Restituisce la grandezza in punti del simbolo corrente nella valuta di quotazione	Verifica Stato
PositionGetDouble	Restituisce la proprietà richiesta di una posizione aperta (double)	Funzioni di Trade
PositionGetInteger	Restituisce la proprietà richiesta di una posizione aperta (datetime o int)	Funzioni di Trade
PositionGetString	Restituisce la proprietà richiesta di una posizione aperta (string)	Funzioni di Trade
PositionGetSymbol	Restituisce il simbolo corrispondente alla posizione aperta	Funzioni di Trade
PositionSelect	Sceglie una posizione aperta per un' ulteriore lavorazione con essa	Funzioni di Trade
PositionsTotal	Restituisce il numero di posizioni aperte	Funzioni di Trade
pow	Aumenta la base alla potenza specificata	Funzioni Matematiche
Print	Mostra un messaggio nel log	Funzioni Comuni
PrintFormat	Formatta e stampa il set di simboli e valori in un file di log secondo il presente formato	Funzioni Comuni
rand	Restituisce un valore pseudocasuale nell'intervallo da 0 a 32767	Funzioni Matematiche
ResetLastError	Imposta a zero il valore di una variabile predeterminata _LastError	Funzioni Comuni
ResourceCreate	Crea una risorsa immagine basata su un set di dati	Funzioni Comuni
ResourceFree	Elimina le risorse create dinamicamente (liberando la memoria allocata per esse)	Funzioni Comuni
ResourceReadImage	Legge i dati dalla risorsa grafica creata dalla funzione	Funzioni Comuni

Funzione	Azione	Section
	ResourceCreate() o salvati nel file EX5 durante la compilazione	
ResourceSav	Salva una risorsa nel file specificato	Funzioni Comuni
round	Arrotonda un valore al numero intero più vicino	Funzioni Matematiche
SendFTP	Invia un file all'indirizzo specificato nella finestra delle impostazioni della scheda "FTP"	Funzioni Comuni
SendMail	Invia una e-mail all'indirizzo specificato nella finestra delle impostazioni della scheda "Email"	Funzioni Comuni
SendNotification	Invia una notifica push ai terminali mobili, il cui ID MetaQuotes è specificato nella scheda "Notifiche"	Funzioni Comuni
SeriesInfoInteger	Restituisce le informazioni sullo stato dei dati storici	Accesso alle Timeseries ed Indicatori
SetIndexBuffer	Associa il buffer indicatore specificato con l' array dinamico uni-dimensionale di tipo double	Indicatori Personalizzati
ShortArrayToString	Copia parte array in una stringa	Funzioni di Conversione
ShortToString	Conversione codice simbolo (unicode) in stringa costituida da un-simbolo	Funzioni di Conversione
SignalBaseGetDouble	Restituisce il valore della proprietà di tipo double per il segnale selezionato	Trade Signals
SignalBaseGetInteger	Restituisce il valore della proprietà di tipo integer per il segnale selezionato	Trade Signals
SignalBaseGetString	Restituisce il valore della proprietà di tipo string per il segnale selezionato	Trade Signals
SignalBaseSelect	Seleziona un segnale dai segnali disponibili nel terminale per ulteriore lavorazione con esso	Trade Signals
SignalBaseTotal	Restituisce la quantità totale di segnali, disponibile nel terminal	Trade Signals

Funzione	Azione	Section
SignalInfoGetDouble	Restituisce il valore della proprietà di tipo double per i parametri di copia del segnale	Trade Signals
SignalInfoGetInteger	Restituisce il valore della proprietà di tipo integer per i parametri di copia del segnale	Trade Signals
SignalInfoGetString	Restituisce il valore della proprietà di tipo string per i parametri di copia del segnale	Trade Signals
SignalInfoSetDouble	Imposta il valore della proprietà di tipo double per i parametri di copia del segnale	Trade Signals
SignalInfoSetInteger	Imposta il valore della proprietà di tipo integer per i parametri di copia del segnale	Trade Signals
SignalSubscribe	Sottoscrive il segnale di trading	Trade Signals
SignalUnsubscribe	Annulla sottoscrizione	Trade Signals
sin	Restituisce il seno di un numero	Funzioni Matematiche
Sleep	Sospende l'esecuzione del corrente Expert Advisor o Script in un intervallo specificato	Funzioni Comuni
SocketCreate	Creare un socket con flag specificati e restituisce il relativo handle	Funzioni di Rete
SocketClose	Chiude un socket	Funzioni di Rete
SocketConnect	Si connette al server con il controllo del timeout	Funzioni di Rete
SocketIsConnected	Controlla se il socket è attualmente connesso	Funzioni di Rete
SocketIsReadable	Ottiene un numero di byte che può essere letto da un socket	Funzioni di Rete
SocketIsWritable	Verifica se i dati possono essere scritti su un socket al momento attuale	Funzioni di Rete
SocketTimeouts	Imposta i timeout per la ricezione e l'invio di dati per un oggetto di sistema socket	Funzioni di Rete
SocketRead	Legge i dati da un socket	Funzioni di Rete

Funzione	Azione	Section
SocketSend	Scrive dati su un socket	Funzioni di Rete
SocketTlsHandshake	Avvia connessione TLS sicura (SSL) verso un host specificato tramite il protocollo TLS Handshake	Funzioni di Rete
SocketTlsCertificate	Ottiene dati sul certificato utilizzato per proteggere la connessione di rete	Funzioni di Rete
SocketTlsRead	Legge i dati dalla connessione TLS protetta	Funzioni di Rete
SocketTlsReadAvailable	Legge tutti i dati disponibili dalla connessione TLS protetta	Funzioni di Rete
SocketTlsSend	Invia dati tramite connessione TLS protetta	Funzioni di Rete
sqrt	Restituisce una radice quadrata	Funzioni Matematiche
srand	Consente di impostare il punto di partenza per la generazione di una serie di interi pseudocasuali	Funzioni Matematiche
StringAdd	Aggiunge una stringa alla fine di un'altra stringa	Funzioni Stringa
StringBufferLen	Restituisce la grandezza del buffer allocato per la stringa	Funzioni Stringa
StringCompare	Confronta due stringhe e restituisce 1 se la prima stringa è maggiore della seconda; 0 - se le stringhe sono uguali; -1 (meno 1) - se la prima stringa è minore della seconda	Funzioni Stringa
StringConcatenate	Forma una serie di parametri passati	Funzioni Stringa
StringFill	Riempie una stringa specificata da simboli selezionati	Funzioni Stringa
StringFind	Ricerca di una sottostringa in una stringa	Funzioni Stringa
StringFormat	Conversione numero in stringa secondo il formato predefinito	Funzioni di Conversione
StringGetCharacter	Restituisce il valore di un numero situato nella posizione specificata della stringa	Funzioni Stringa

Funzione	Azione	Section
StringInit	Inizializza la stringa per simboli specificati e fornisce la lunghezza della stringa specificata	Funzioni Stringa
StringLen	Restituisce il numero di simboli in una stringa	Funzioni Stringa
StringReplace	Sostituisce tutte le sottostringhe trovate di una stringa da una sequenza di set di simboli	Funzioni Stringa
StringSetCharacter	Restituisce una copia di una stringa con un valore modificato di un simbolo in una posizione specificata	Funzioni Stringa
StringSplit	Ottiene sottostringhe da un separatore specificato dalla stringa specificata, restituisce il numero di sottostringhe ottenute	Funzioni Stringa
StringSubstr	Estrae una sottostringa da una stringa di testo a partire da una posizione specificata	Funzioni Stringa
StringToCharArray	Copia senso-simbolo di una string convertita da Unicode ad ANSI, in un luogo selezionato di array di tipo uchar	Funzioni di Conversione
StringToColor	Conversione stringa "R,G,B" o una stringa con il nome del colore in valore di tipo di colore	Funzioni di Conversione
StringToDouble	La conversione di una stringa contenente una rappresentazione simbolo di numero in numero di tipo double	Funzioni di Conversione
StringToInteger	Conversione di una stringa contenente una rappresentazione di un simbolo di numero in numero di tipo int	Funzioni di Conversione
StringToLower	Trasforma tutti i simboli di una stringa selezionata in minuscolo per posizione	Funzioni Stringa
StringToShortArray	Copia in senso-simbolo di una una stringa ad una parte dell' array selezionato di tipo ushort	Funzioni di Conversione

Funzione	Azione	Section
StringToTime	Conversione di una stringa contenente l'ora o la data in formato di tipo datetime "aaaa.mm.gg [oo:mi]"	Funzioni di Conversione
StringToUpper	Trasforma tutti i simboli di una stringa selezionata in maiuscole, per posizione	Funzioni Stringa
StringTrimLeft	Taglia caratteri di alimentazione di linea, spazi e le schede nella parte sinistra della stringa	Funzioni Stringa
StringTrimRight	Taglia i caratteri di alimentazione di linea, spazi e le schede nella parte destra della stringa	Funzioni Stringa
StructToTime	Converte una variabile di tipo struttura MqlDateTime in un valore datetime	Data ed Ora
Symbol	Restituisce il nome del simbolo del corrente grafico	Verifica Stato
SymbolInfoDouble	Restituisce il valore double del simbolo per la proprietà corrispondente	Market Info
SymbolInfoInteger	Restituisce un valore di tipo integer (long, datetime, int o bool) di un simbolo specificato per la proprietà corrispondente	Market Info
SymbolInfoMarginRate	Returns the margin rates depending on the order type and direction	Market Info
SymbolInfoSessionQuote	Permette di ricevere l'ora di inizio e fine delle sessioni quotate specificate per un simbolo e giorno della settimana specifici.	Market Info
SymbolInfoSessionTrade	Permette di ricevere l'ora di inizio e fine delle sessioni di trade specificate per un simbolo e giorno della settimana specifici.	Market Info
SymbolInfoString	Restituisce un valore del tipo di string di un simbolo specificato per la proprietà corrispondente	Market Info
SymbolInfoTick	Restituisce i prezzi correnti per il simbolo di cui una variabile di	Market Info

Funzione	Azione	Section
	tipo MqlTick	
SymbolIsSynchronized	Controlla se i dati di un simbolo selezionato nel terminale sono sincronizzati con i dati relativi al trade server	Market Info
SymbolName	Restituisce il nome di un simbolo specificato	Market Info
SymbolSelect	Consente di selezionare un simbolo nella finestra di controllo del mercato o rimuove un simbolo dalla finestra	Market Info
SymbolsTotal	Restituisce il numero di simboli disponibili (selezionati nel Market Watch o tutti)	Market Info
tan	Restituisce la tangente di un numero	Funzioni Matematiche
TerminalClose	Comanda al terminale di completare l'operazione	Funzioni Comuni
TerminalInfoDouble	Restituisce un valore double di una proprietà corrispondente dell'ambiente programma MQL5	Verifica Stato
TerminalInfoInteger	Restituisce un valore integer della corrispondente proprietà dell'ambiente del programma mql5.	Verifica Stato
TerminalInfoString	Restituisce il valore stringa della corrispondente proprietà dell'ambiente del programma mql5.	Verifica Stato
TesterStatistics	Restituisce il valore di una statistica specificata calcolata in base ai risultati di testing	Funzioni Comuni
TextGetSize	Restituisce la larghezza della stringa e l'altezza alle correnti Impostazioni dei caratteri	Funzioni Oggetto
TextOut	Trasferisce il testo ad un array personalizzato (buffer) progettato per la creazione di una risorsa grafica	Funzioni Oggetto
TextSetFont	Consente di impostare il tipo di carattere per la visualizzazione	Funzioni Oggetto

Funzione	Azione	Section
	del testo utilizzando metodi di disegno (Arial 20 utilizzato per impostazione predefinita)	
TimeCurrent	Restituisce l'ultimo orario conosciuto di server (orario della ricevuta dell'ultima quotazione) nel formato datetime	Data ed Ora
TimeDaylightSavings	Restituisce il segno di switch dell'ora legale	Data ed Ora
TimeGMT	Restituisce GMT in formato datetime con l'ora legale dell' orario locale del computer, in cui il terminale client è in esecuzione	Data ed Ora
TimeGMTOffset	Restituisce la corrente differenza tra l'orario GMT e l'ora del computer locale in secondi, tenendo in considerazione l'interruttore DST	Data ed Ora
TimeLocal	Restituisce l'ora locale del computer in formato datetime	Data ed Ora
TimeToString	Conversione di un valore contenente il tempo in secondi trascorsi dal 01.01.1970 in una stringa di formato "aaaa.mm.gg. oo:mi" formato	Funzioni di Conversione
TimeToStruct	Converte un valore datetime in una variabile di tipo struttura MqlDateTime	Data ed Ora
TimeTradeServer	Restituisce l'ora attuale del calcolo del server di trade	Data ed Ora
UninitializeReason	Restituisce il codice della ragione per la deinizializzazione	Verifica Stato
WebRequest	Invia la richiesta HTTP al server specificato	Funzioni Comuni
ZeroMemory	Resetta la variabile passata ad esso per riferimento. La variabile può essere di qualsiasi tipo, eccetto per le classi e strutture che hanno costruttori.	Funzioni Comuni

List of MQL5 Constants

All MQL5 constants in alphabetical order.

Constant	Descrizione	Usage
__DATE__	Compilazione di file senza orario (ore, minuti e secondi sono uguali a 0)	Print
__DATETIME__	Data ed ora di compilazione del file	Print
__FILE__	Nome del file attualmente elaborati	Print
__FUNCSIG__	Firma della funzione nel cui corpo si trova la macro. Logging della descrizione completa delle funzioni può essere utile per l'identificazione di funzioni	Print

Constant	Descrizione	Usage
	sovraccaricate	
__FUNCTION__	Nome della funzione, nel cui corpo si trova la macro	Print
__LINE__	Numero di riga nel codice sorgente, in cui si trova la macro	Print
__MQLBUILD__, __MQL5BUILD__	Numero di build compilatore	Print
__PATH__	Un percorso assoluto del file che è attualmente in fase di compilazione	Print
ACCOUNT_ASSETS	The current assets of an account	AccountInfoDouble
ACCOUNT_BALANCE	Bilancio dell'account in valuta di deposito	AccountInfoDouble
ACCOUNT_COMMISSION_BLOCKED	The current	AccountInfoDouble

Constant	Descrizione	Usage
	blocked commission amount on an account	
ACCOUNT_COMPANY	Nome della società che fornisce l'account	AccountInfoString
ACCOUNT_CREDIT	Credito dell'account nella valuta di deposito	AccountInfoDouble
ACCOUNT_CURRENCY	Valuta dell'account	AccountInfoString
ACCOUNT_EQUITY	Equità nella valuta di deposito	AccountInfoDouble
ACCOUNT_LEVERAGE	Leverage dell'account	AccountInfoInteger
ACCOUNT_LIABILITIES	The current liabilities on an account	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	Numero massimo consentito di ordini pendenti attivi	AccountInfoInteger
ACCOUNT_LOGIN	Numero di account	AccountInfoInteger

Constant	Descrizione	Usage
ACCOUNT_MARGIN	Margine utilizzzato dell'account nella valuta di deposito	AccountInfoDouble
ACCOUNT_MARGIN_FREE	Margine libero dell'account nella valuta di deposito	AccountInfoDouble
ACCOUNT_MARGIN_INITIAL	Initial margin. The amount reserved on an account to cover the margin of all pending orders	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	Livello di margine dell'account in percentuale	AccountInfoDouble
ACCOUNT_MARGIN_MAINTENANCE	Maintenance margin. The minimum equity reserved on an account to cover the minimu	AccountInfoDouble

Constant	Descrizione	Usage
	m amount of all open positions	
ACCOUNT_MARGIN_SO_CALL	Livello di margin call. A seconda dell'impostazione ACCOUNT_MARGIN_SO_MODE è espressa in percentuale o in valuta di deposito	AccountInfoDouble
ACCOUNT_MARGIN_SO_MODE	Modalità per l'impostazione del margine minimo consentito	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	Livello di stop out del margine. A seconda dell'impostazione ACCOUNT_MARGIN_SO_MODE è espressa in percentu	AccountInfoDouble

Constant	Descrizione	Usage
	ale o in valuta di deposito	
ACCOUNT_NAME	Nome del cliente	AccountInfoString
ACCOUNT_PROFIT	Profitto attuale dell'account in valuta di deposito	AccountInfoDouble
ACCOUNT_SERVER	Nome del trade server	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	Modalità di stop out dell'account in termini monetari	AccountInfoInteger
ACCOUNT_STOPOUT_MODE_PERCENT	Modalità di stop out dell' account in termini percentuali	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	Trade consentito per il corrente account	AccountInfoInteger
ACCOUNT_TRADE_EXPERT	Trade consentito per un Expert Advisor	AccountInfoInteger

Constant	Descrizione	Usage
ACCOUNT_TRADE_MODE	Modalità trade dell' Account	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	Account contest	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	Account demo	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	Account reale	AccountInfoInteger
ALIGN_CENTER	Centrato (solo per l'oggetto Edit)	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_LEFT	L'allineamento a sinistra	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	Allineamento a destra	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	Punto di ancoraggio strettamente nel centro dell'oggetto	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT	Punto di ancoraggio a sinistra nel centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_LOWER	Punto di ancoraggio nell'angolo in basso a sinistra	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
ANCHOR_LEFT_UPPER	Punto di ancoraggio in alto a sinistra	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	Punto di ancoraggio sotto sotto nel centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	Punto di ancoraggio a destra al centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_LOWER	Punto di ancoraggio nell'angolo in basso a destra	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_UPPER	Punto di ancoraggio in alto a destra	ObjectSetInteger, ObjectGetInteger
ANCHOR_UPPER	Punto di ancoraggio sopra nel centro	ObjectSetInteger, ObjectGetInteger
BASE_LINE	Linea principale	Linee Indicatori
BOOK_TYPE_BUY	Ordine di Buy (Acquisto)	MqlBookInfo
BOOK_TYPE_BUY_MARKET	Ordine di Buy	MqlBookInfo

Constant	Descrizione	Usage
	dal Mercato	
BOOK_TYPE_SELL	Ordine di Sell(Offerta)	MqlBookInfo
BOOK_TYPE_SELL_MARKET	Ordine di Sell dal Mercato	MqlBookInfo
BORDER_FLAT	Forma piatta	ObjectSetInteger, ObjectGetInteger
BORDER_RAISED	Forma sporgente	ObjectSetInteger, ObjectGetInteger
BORDER_SUNKEN	Forma concava	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	Valore massimo, che può essere rappresentato dal tipo char	Costanti di tipo numerico
CHAR_MIN	Valore minimo, che può essere rappresentato dal tipo char	Costanti di tipo numerico
CHART_AUTOSCROLL	Modalità automatica di spostamento al bordo destro del grafico	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
CHART_BARS	Visualizzare come una sequenza di barre	ChartSetInteger
CHART_BEGIN	Inizio del Grafico (i prezzi più vecchi)	ChartNavigate
CHART_BRING_TO_TOP	Vede il grafico sopra altri grafici	ChartSetInteger, ChartGetInteger
CHART_CANDLES	Visualizza come candele giapponesi	ChartSetInteger
CHART_COLOR_ASK	Colore del livello di prezzo Ask	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	Colore di sfondo del Grafico	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	Colore del livello di prezzo Bid	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	Colore del corpo della candela bear	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
CHART_COLOR_CANDLE_BULL	Colore del corpo della candela toro	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	Colore per la barra in alto, ombre bordi del corpo di candele bear	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	Colore linea chart e colore della candela giapponese "Doji"	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	Colore per la barra in alto, ombre bordi del corpo di candele bull	ChartSetInteger, ChartGetInteger
CHART_COLOR_FOREGROUND	Colore degli assi, scale e linea OHLC	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	Colore della griglia	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	Colore della	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
	linea dell'ultimo prezzo affare eseguito (Last)	
CHART_COLOR_STOP_LEVEL	Colore dei livelli di ordini di stop (Stop Loss e Take Profit)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	Colore dei volumi e livelli di apertura della posizione	ChartSetInteger, ChartGetInteger
CHART_COMMENT	Commento di testo in un grafico	ChartSetString, ChartGetString
CHART_CURRENT_POS	Posizione attuale	ChartNavigate
CHART_DRAG_TRADE_LEVELS	Il permesso di trascinare i livelli di trading su un grafico con il mouse. La modalità di trascina	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
	mento è attivata per impostazione predefinita (valore true)	
CHART_EVENT_MOUSE_MOVE	Invia notifiche di movimento del mouse ed eventi click del mouse (CHART_EVENT_MOUSE_MOVE) a tutti i programmi MQL5 programmi su un grafico	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_CREATE	Inviare la notifica di un evento di creazione del nuovo oggetto (CHART_EVENT_OBJECT_CREATE) a tutti i programmi mql5	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
	sul grafico	
CHART_EVENT_OBJECT_DELETE	Inviare la notifica di un evento di eliminazione oggetto (CHART_EVENT_OBJECT_DELETE) a tutti i programmi mql5 sul grafico	ChartSetInteger, ChartGetInteger
CHART_FIRST_VISIBLE_BAR	Numero della prima barra visibile nel grafico. L'indicizzazione delle barre è uguale per TimeSeries.	ChartSetInteger, ChartGetInteger
CHART_FIXED_MAX	Fixed chart maximum	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	Minimo fisso del Grafico	ChartSetDouble, ChartGetDouble
CHART_FIXED_POSITION	Posizione fissa dal	ChartSetDouble, ChartGetDouble

Constant	Descrizione	Usage
	bordo sinistro in valore percentuale. La posizione fissa del grafico è contrassegnata da un piccolo triangolo grigio sull'asse del tempo orizzontale. Viene visualizzato solo se lo slittamento automatico del grafico a destra sul segno di spunta in entrata è disattivato (vedere la proprietà CHART_AUTOSCROLL). La barra in una posizion	

Constant	Descrizione	Usage
	e fissa rimane nello stesso posto durante lo zoom in e out.	
CHART_FOREGROUND	Grafico dei prezzi in primo piano	ChartSetInteger, ChartGetInteger
CHART_HEIGHT_IN_PIXELS	Altezza del grafico in pixel	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	Identifica l'oggetto "Chart" (OBJ_CHART) oggetto - restituisce true per un oggetto grafico. Restituisce false per un grafico vero e proprio	ChartSetInteger, ChartGetInteger
CHART_LINE	Visualizza come una linea tracciata dai prezzi Close	ChartSetInteger

Constant	Descrizione	Usage
CHART_MODE	Tipo di grafico (candele, barre o di linea)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	Scorre la tabella utilizzando orizzontalmente il tasto sinistro del mouse. Lo scrolling verticale è disponibile anche se il valore di ogni seguente proprietà è impostato su true: CHART_SCALEFIX, CHART_SCALEFIX_11 o CHART_SCALE_PT_PER_BAR	ChartSetInteger, ChartGetInteger
CHART_POINTS_PER_BAR	Scala in punti per barra	ChartSetDouble, ChartGetDouble

Constant	Descrizione	Usage
CHART_PRICE_MAX	Massimo del grafico	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	Minimo del Grafico	ChartSetDouble, ChartGetDouble
CHART_SCALE	Scale	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	Scala da specificare in punti per barre	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	Modalità scala fissa	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX_11	Modalità Scala 1:1	ChartSetInteger, ChartGetInteger
CHART_SHIFT	Modalità di grafico dei prezzi indentata dal bordo destro	ChartSetInteger, ChartGetInteger
CHART_SHIFT_SIZE	La grandezza della barra zero indentata dal bordo destro, in percentuale	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	Mostra valori Ask	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
	come una linea orizzontale nel grafico	
CHART_SHOW_BID_LINE	Mostra i valori Bid come una linea orizzontale nel grafico	ChartSetInteger, ChartGetInteger
CHART_SHOW_DATE_SCALE	Risultati della scala temporale sul grafico	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	Visualizza griglia nel grafico	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	Visualizza valori Last come una linea orizzontale in un grafico	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	Descrizioni Pop-up di oggetti grafici	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	Mostra valori OHLC nell'angolo in alto	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
	a sinistra	
CHART_SHOW_ONE_CLICK	Showing the "One click trading" panel on a chart	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	Visualizza separatori verticali tra periodi adiacenti	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	Risultati della scala dei prezzi sul grafico	ChartSetInteger, ChartGetInteger
CHART_SHOW_TRADE_LEVELS	Visualizzazione dei livelli di trade nel grafico (livelli di posizioni aperte, Stop Loss, Take Profit ed ordini pendenti)	ChartSetInteger, ChartGetInteger
CHART_SHOW_VOLUMES	Visualizzare volume nel grafico	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
CHART_VISIBLE_BARS	Il numero di barre sul grafico che può essere visualizzato	ChartSetInteger, ChartGetInteger
CHART_VOLUME_HIDE	I volumi non sono mostrati	ChartSetInteger
CHART_VOLUME_REAL	Volumi Trade	ChartSetInteger
CHART_VOLUME_TICK	Volumi Tick	ChartSetInteger
CHART_WIDTH_IN_BARS	Larghezza del grafico in barre	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	Larghezza grafico in pixel	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	Handle della finestra del grafico (HWND)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	Visibilità delle sottofinestre	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	La distanza tra il frame superiore della sottofinestra indicatore ed il	ChartSetInteger, ChartGetInteger

Constant	Descrizione	Usage
	frame superiore della finestra grafico principale, lungo l'asse verticale Y, in pixel. In caso di un evento del mouse, le coordinate del cursore vengono passate in termini di coordinate della finestra grafico principale, mentre le coordinate di oggetti grafici in una finestra indicatore secondaria vengono impostati rispetto all'angol	

Constant	Descrizione	Usage
	o in alto a sinistra della finestra secondaria. Il valore è necessario per convertire le coordinate assolute del grafico principale alle coordinate locali della sottofinestra per il corretto lavoro con gli oggetti grafici, le cui coordinate sono impostate rispetto all'angolo superiore sinistro del frame sottofinestra.	

Constant	Descrizione	Usage
CHART_WINDOWS_TOTAL	Il numero totale di finestre del grafico, tra cui sottofinestre indicatore	ChartSetInteger, ChartGetInteger
CHARTEVENT_CHART_CHANGE	Modifica della grandezza del grafico o modifica delle proprietà del grafico tramite la finestra delle Proprietà	OnChartEvent
CHARTEVENT_CLICK	Facendo clic su un grafico	OnChartEvent
CHARTEVENT_CUSTOM	Numero iniziale di un evento da una serie di eventi personalizzati	OnChartEvent
CHARTEVENT_CUSTOM_LAST	Il numero finale di un evento	OnChartEvent

Constant	Descrizione	Usage
	di una serie di eventi personalizzati	
CHARTEVENT_KEYDOWN	Tasti	OnChartEvent
CHARTEVENT_MOUSE_MOVE	Movimento del mouse, clic del mouse (se CHART_EVENT_MOUSE_MOVE=true è impostato per il grafico)	OnChartEvent
CHARTEVENT_OBJECT_CHANGE	Oggetto grafico proprietà modificata tramite la finestra delle proprietà	OnChartEvent
CHARTEVENT_OBJECT_CLICK	Facendo clic su un oggetto grafico	OnChartEvent
CHARTEVENT_OBJECT_CREATE	Oggetto grafico creato (se CHART_EVENT_OBJECT_CREATE	OnChartEvent

Constant	Descrizione	Usage
	=true è impostato per il grafico)	
CHARTEVENT_OBJECT_DELETE	Oggetto grafico eliminato (se CHART_EVENT_OBJECT_DELETE=true è impostato per il grafico)	OnChartEvent
CHARTEVENT_OBJECT_DRAG	Drag and drop di un oggetto grafico	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	Fine di modifica del testo nella modifica degli oggetti grafici	OnChartEvent
CHARTS_MAX	Il numero massimo possibile di grafici aperti simultaneamente nel terminale	Altre costanti
CHIKOUSPAN_LINE	Linea Chikou Span	Linee Indicatori

Constant	Descrizione	Usage
clrAliceBlue	Alice Blue	I Web Colors
clrAntiqueWhite	Antique White	I Web Colors
clrAqua	Aqua	I Web Colors
clrAquamarine	Aquamarine	I Web Colors
clrBeige	Beige	I Web Colors
clrBisque	Bisque	I Web Colors
clrBlack	Black	I Web Colors
clrBlanchedAlmond	Blanched Almond	I Web Colors
clrBlue	Blue	I Web Colors
clrBlueViolet	Blue Violet	I Web Colors
clrBrown	Brown	I Web Colors
clrBurlyWood	Burly Wood	I Web Colors
clrCadetBlue	Cadet Blue	I Web Colors
clrChartreuse	Chartreuse	I Web Colors
clrChocolate	Chocolate	I Web Colors
clrCoral	Coral	I Web Colors
clrCornflowerBlue	Cornflower Blue	I Web Colors
clrCornsilk	Cornsilk	I Web Colors
clrCrimson	Crimson	I Web Colors
clrDarkBlue	Dark Blue	I Web Colors
clrDarkGoldenrod	Dark Goldenrod	I Web Colors

Constant	Descrizione	Usage
clrDarkGray	Dark Gray	I Web Colors
clrDarkGreen	Dark Green	I Web Colors
clrDarkKhaki	Dark Khaki	I Web Colors
clrDarkOliveGreen	Dark Olive Green	I Web Colors
clrDarkOrange	Dark Orange	I Web Colors
clrDarkOrchid	Dark Orchid	I Web Colors
clrDarkSalmon	Dark Salmon	I Web Colors
clrDarkSeaGreen	Dark Sea Green	I Web Colors
clrDarkSlateBlue	Dark Slate Blue	I Web Colors
clrDarkSlateGray	Dark Slate Gray	I Web Colors
clrDarkTurquoise	Dark Turquoise	I Web Colors
clrDarkViolet	Dark Violet	I Web Colors
clrDeepPink	Deep Pink	I Web Colors
clrDeepSkyBlue	Deep Sky Blue	I Web Colors
clrDimGray	Dim Gray	I Web Colors
clrDodgerBlue	Dodger Blue	I Web Colors
clrFireBrick	Fire Brick	I Web Colors

Constant	Descrizione	Usage
clrForestGreen	Forest Green	I Web Colors
clrGainsboro	Gainsboro	I Web Colors
clrGold	Gold	I Web Colors
clrGoldenrod	Goldenrod	I Web Colors
clrGray	Gray	I Web Colors
clrGreen	Green	I Web Colors
clrGreenYellow	Green Yellow	I Web Colors
clrHoneydew	Honeydew	I Web Colors
clrHotPink	Hot Pink	I Web Colors
clrIndianRed	Indian Red	I Web Colors
clrIndigo	Indigo	I Web Colors
clrIvory	Ivory	I Web Colors
clrKhaki	Khaki	I Web Colors
clrLavender	Lavender	I Web Colors
clrLavenderBlush	Lavender Blush	I Web Colors
clrLawnGreen	Lawn Green	I Web Colors
clrLemonChiffon	Lemon Chiffon	I Web Colors
clrLightBlue	Light Blue	I Web Colors
clrLightCoral	Light Coral	I Web Colors
clrLightCyan	Light Cyan	I Web Colors
clrLightGoldenrod	Light Goldenro	I Web Colors

Constant	Descrizione	Usage
	d	
clrLightGray	Light Gray	I Web Colors
clrLightGreen	Light Green	I Web Colors
clrLightPink	Light Pink	I Web Colors
clrLightSalmon	Light Salmon	I Web Colors
clrLightSeaGreen	Light Sea Green	I Web Colors
clrLightSkyBlue	Light Sky Blue	I Web Colors
clrLightSlateGray	Light Slate Gray	I Web Colors
clrLightSteelBlue	Light Steel Blue	I Web Colors
clrLightYellow	Light Yellow	I Web Colors
clrLime	Lime	I Web Colors
clrLimeGreen	Lime Green	I Web Colors
clrLinen	Linen	I Web Colors
clrMagenta	Magenta	I Web Colors
clrMaroon	Maroon	I Web Colors
clrMediumAquamarine	Medium Aquamarine	I Web Colors
clrMediumBlue	Medium Blue	I Web Colors
clrMediumOrchid	Medium Orchid	I Web Colors
clrMediumPurple	Medium Purple	I Web Colors

Constant	Descrizione	Usage
clrMediumSeaGreen	Medium Sea Green	I Web Colors
clrMediumSlateBlue	Medium Slate Blue	I Web Colors
clrMediumSpringGreen	Medium Spring Green	I Web Colors
clrMediumTurquoise	Medium Turquoise	I Web Colors
clrMediumVioletRed	Medium Violet Red	I Web Colors
clrMidnightBlue	Midnight Blue	I Web Colors
clrMintCream	Mint Cream	I Web Colors
clrMistyRose	Misty Rose	I Web Colors
clrMoccasin	Moccasin	I Web Colors
clrNavajoWhite	Navajo White	I Web Colors
clrNavy	Navy	I Web Colors
clrNONE	Assenza di colore	Altre costanti
clrOldLace	Old Lace	I Web Colors
clrOlive	Olive	I Web Colors
clrOliveDrab	Olive Drab	I Web Colors
clrOrange	Orange	I Web Colors
clrOrangeRed	Orange Red	I Web Colors
clrOrchid	Orchid	I Web Colors

Constant	Descrizione	Usage
clrPaleGoldenrod	Pale Goldenrod	I Web Colors
clrPaleGreen	Pale Green	I Web Colors
clrPaleTurquoise	Pale Turquoise	I Web Colors
clrPaleVioletRed	Pale Violet Red	I Web Colors
clrPapayaWhip	Papaya Whip	I Web Colors
clrPeachPuff	Peach Puff	I Web Colors
clrPeru	Peru	I Web Colors
clrPink	Pink	I Web Colors
clrPlum	Plum	I Web Colors
clrPowderBlue	Powder Blue	I Web Colors
clrPurple	Purple	I Web Colors
clrRed	Red	I Web Colors
clrRosyBrown	Rosy Brown	I Web Colors
clrRoyalBlue	Royal Blue	I Web Colors
clrSaddleBrown	Saddle Brown	I Web Colors
clrSalmon	Salmon	I Web Colors
clrSandyBrown	Sandy Brown	I Web Colors
clrSeaGreen	Sea Green	I Web Colors
clrSeashell	Seashell	I Web Colors
clrSienna	Sienna	I Web Colors

Constant	Descrizione	Usage
clrSilver	Silver	I Web Colors
clrSkyBlue	Sky Blue	I Web Colors
clrSlateBlue	Slate Blue	I Web Colors
clrSlateGray	Slate Gray	I Web Colors
clrSnow	Snow	I Web Colors
clrSpringGreen	Spring Green	I Web Colors
clrSteelBlue	Steel Blue	I Web Colors
clrTan	Tan	I Web Colors
clrTeal	Teal	I Web Colors
clrThistle	Thistle	I Web Colors
clrTomato	Tomato	I Web Colors
clrTurquoise	Turquoise	I Web Colors
clrViolet	Violet	I Web Colors
clrWheat	Wheat	I Web Colors
clrWhite	White	I Web Colors
clrWhiteSmoke	White Smoke	I Web Colors
clrYellow	Yellow	I Web Colors
clrYellowGreen	Yellow Green	I Web Colors
CORNER_LEFT_LOWER	Il centro delle coordinate è nell'angolo in basso a sinistra del grafico	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
CORNER_LEFT_UPPER	Il centro delle coordinate è nell'angolo superiore sinistro del grafico	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_LOWER	Il centro di coordinate è nell'angolo in basso a destra del grafico	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_UPPER	Il centro di coordinate è nell'angolo in alto a destra del grafico	ObjectSetInteger, ObjectGetInteger
CP_ACP	L'attuale codepage Windows ANSI	CharArrayToString, StringToCharArray, FileOpen
CP_MACCP	L'attuale code page di sistema Macintosh. Nota: Questo valore è usato	CharArrayToString, StringToCharArray, FileOpen

Constant	Descrizione	Usage
	soprattutto nei codici di programma creati prima, e non serve a niente ora, dal momento che i moderni computer Macintosh usano Unicode per la codifica.	
CP_OEMCP	L'attuale sistema OEM code page.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	Simbolo codice della pagina	CharArrayToString, StringToCharArray, FileOpen
CP_THREAD_ACP	La tabella codici ANSI di Windows per il thread corrente.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF7	UTF-7 code page.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	UTF-8 code page.	CharArrayToString, StringToCharArray, FileOpen

Constant	Descrizione	Usage
CRYPT_AES128	AES encryption with 128 bit key (16 bytes)	CryptEncode, CryptDecode
CRYPT_AES256	AES encryption with 256 bit key (32 bytes)	CryptEncode, CryptDecode
CRYPT_ARCH_ZIP	ZIP archives	CryptEncode, CryptDecode
CRYPT_BASE64	BASE64	CryptEncode, CryptDecode
CRYPT_DES	DES encryption with 56 bit key (7 bytes)	CryptEncode, CryptDecode
CRYPT_HASH_MD5	MD5 HASH caculation	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	SHA1 HASH caculation	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	SHA256 HASH caculation	CryptEncode, CryptDecode
DBL_DIG	Numero di cifre decimali significative per il tipo double	Costanti di tipo numerico
DBL_EPSILON	Valore minimo,	Costanti di tipo numerico

Constant	Descrizione	Usage
	che soddisfa la condizione: 1.0+DBL_EPSILON != 1.0 (per il tipo double)	
DBL_MANT_DIG	Conteggio bits in una mantissa per il tipo double	Costanti di tipo numerico
DBL_MAX	Valore massimo, che può essere rappresentato dal tipo double	Costanti di tipo numerico
DBL_MAX_10_EXP	Massimo valore decimale del grado dell'esponente per il tipo double	Costanti di tipo numerico
DBL_MAX_EXP	Massimo valore binario del grado dell'esponente per il	Costanti di tipo numerico

Constant	Descrizione	Usage
	tipo double	
DBL_MIN	Valore minimo, che può essere rappresentato dal tipo double	Costanti di tipo numerico
DBL_MIN_10_EXP	Minimo valore decimale del grado dell'esponente per il tipo double	Costanti di tipo numerico
DBL_MIN_EXP	Minimo valore binario del grado dell'esponente per il tipo double	Costanti di tipo numerico
DEAL_COMMENT	Commento affare	HistoryDealGetString
DEAL_COMMISSION	Prezzo commissione	HistoryDealGetDouble
DEAL_ENTRY	Entry dell'affare - ingresso in, ingresso out, inversione	HistoryDealGetInteger

Constant	Descrizione	Usage
DEAL_ENTRY_IN	Entry in	HistoryDealGetInteger
DEAL_ENTRY_INOUT	Reverse	HistoryDealGetInteger
DEAL_ENTRY_OUT	Entry out	HistoryDealGetInteger
DEAL_MAGIC	Numero magico dell' affare (vedi ORDER_MAGIC)	HistoryDealGetInteger
DEAL_ORDER	Affare numero d'ordine	HistoryDealGetInteger
DEAL_POSITION_ID	Identificatore di posizione, nell' apertura, modifica o cambio di ciò a cui questo affare prende parte. Ogni posizione ha un identificatore univoco che viene assegnato a tutti gli affari eseguiti per il simbolo durante l'intero	HistoryDealGetInteger

Constant	Descrizione	Usage
	ciclo di vita della posizione.	
DEAL_PRICE	Prezzo affare	HistoryDealGetDouble
DEAL_PROFIT	Profitto affare	HistoryDealGetDouble
DEAL_SWAP	Swap cumulativo sulla chiusura	HistoryDealGetDouble
DEAL_SYMBOL	Simbolo affare	HistoryDealGetString
DEAL_TIME	Orario affare	HistoryDealGetInteger
DEAL_TIME_MSC	Il tempo di esecuzione di un affare in millisecondi dal 01.01.1970	HistoryDealGetInteger
DEAL_TYPE	Deal type	HistoryDealGetInteger
DEAL_TYPE_BALANCE	Balance	HistoryDealGetInteger
DEAL_TYPE_BONUS	Bonus	HistoryDealGetInteger
DEAL_TYPE_BUY	Buy	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	Affare buy annullato Non ci può essere una situazione in cui un	HistoryDealGetInteger

Constant	Descrizione	Usage
	affare precedentemente eseguito venga annullato. In questo caso, il tipo di affare precedentemente eseguito (DEAL_TYPE_BUY) viene cambiato in DEAL_TYPE_BUY_CANCELED, ed il suo profitto/perdita viene reso a zero. Il precedente profitto/perdita ottenuti vengono caricati/prelevati usando un'operazione di bilanciamento separata.	

Constant	Descrizione	Usage
DEAL_TYPE_CHARGE	Carico aggiuntivo	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	Commissione aggiuntiva	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	Commissione agente giornaliero	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Commissione agente mensile	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	Commissione giornaliera	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	Commissione mensile	HistoryDealGetInteger
DEAL_TYPE_CORRECTION	Correzione	HistoryDealGetInteger
DEAL_TYPE_CREDIT	Credit	HistoryDealGetInteger
DEAL_TYPE_INTEREST	Tasso d'interesse	HistoryDealGetInteger
DEAL_TYPE_SELL	Sell	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	Affare vendita eseguito. Non ci può essere una situazione in cui un affare di	HistoryDealGetInteger

Constant	Descrizione	Usage
	vendita eseguito in precedenza venga cancellato. In questo caso, il tipo di affare precedentemente eseguito (DEAL_TYPE_SELL) viene cambiato in DEAL_TYPE_SELL_CANCELED, ed il suo profitto/perdita viene reso a zero. Il precedente profitto/perdita ottenuti vengono caricati/prelevati usando un' operazione di bilanciamento separata.	

Constant	Descrizione	Usage
DEAL_VOLUME	Volume affare	HistoryDealGetDouble
DRAW_ARROW	Disegnamento frecce	Stili di Disegno
DRAW_BARS	Visualizzare come una sequenza di barre	Stili di Disegno
DRAW_CANDLES	Visualizza come una sequenza di candele	Stili di Disegno
DRAW_COLOR_ARROW	Disegno frecce multicolori	Stili di Disegno
DRAW_COLOR_BARS	Multicolore barre	Stili di Disegno
DRAW_COLOR_CANDLES	Candele multicolore	Stili di Disegno
DRAW_COLOR_HISTOGRAM	Istogramma multicolore dalla linea dello zero	Stili di Disegno
DRAW_COLOR_HISTOGRAM2	Istogramma multicolore dei due buffer indicatore	Stili di Disegno

Constant	Descrizione	Usage
DRAW_COLOR_LINE	Linea multicolore	Stili di Disegno
DRAW_COLOR_SECTION	Sezione multicolore	Stili di Disegno
DRAW_COLOR_ZIGZAG	ZigZag multicolore	Stili di Disegno
DRAW_FILLING	Color fill between the two levels	Stili di Disegno
DRAW_HISTOGRAM	Istogramma dalla linea dello zero	Stili di Disegno
DRAW_HISTOGRAM2	Istogramma dei due buffer indicatore	Stili di Disegno
DRAW_LINE	Linea	Stili di Disegno
DRAW_NONE	Non disegnato	Stili di Disegno
DRAW_SECTION	Sezione	Stili di Disegno
DRAW_ZIGZAG	Stili Zigzag permettono la sezione verticale sulla barra	Stili di Disegno
ELLIOTT_CYCLE	Ciclo	ObjectSetInteger, ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	Grand Supercycle	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
ELLIOTT_INTERMEDIATE	Intermedio	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	Minore	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	Minuette	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUTE	Minuto	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	Primario	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUBMINUETTE	Subminuette	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUPERCYCLE	Supercycle	ObjectSetInteger, ObjectGetInteger
EMPTY_VALUE	Valore vuoto in un buffer di indicatore	Altre costanti
ERR_ACCOUNT_WRONG_PROPERTY	Proprietà ID dell'account, sbagliata	GetLastError
ERR_ARRAY_BAD_SIZE	La grandezza della grandezza dell'array supera i 2 GB	GetLastError
ERR_ARRAY_RESIZE_ERROR	Memoria insufficiente per il trasferimento di un array, o un tentativo di	GetLastError

Constant	Descrizione	Usage
	modificare la grandezza di un array statico	
ERR_BOOKS_CANNOT_ADD	Profondità di Mercato non può essere aggiunta	GetLastError
ERR_BOOKS_CANNOT_DELETE	Profondità di Mercato non può essere rimossa	GetLastError
ERR_BOOKS_CANNOT_GET	I dati dalla Profondità di Mercato non possono essere ottenuti	GetLastError
ERR_BOOKS_CANNOT_SUBSCRIBE	Errore nella sottoscrizione di ricezione di nuovi dati da Profondità di Mercato	GetLastError
ERR_BUFFERS_NO_MEMORY	Memoria insufficiente per la distribuzione dei buffer	GetLastError

Constant	Descrizione	Usage
	indicatore	
ERR_BUFFERS_WRONG_INDEX	Errato indice buffer indicatore	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	Impossibile cancellare la directory (probabilmente uno o più file sono bloccati e l'operazione di rimozione è fallita)	GetLastError
ERR_CANNOT_DELETE_DIRECTORY	La directory non può essere rimossa	GetLastError
ERR_CANNOT_DELETE_FILE	Errore durante l'eliminazione del file	GetLastError
ERR_CANNOT_OPEN_FILE	Errore durante l'apertura del file	GetLastError
ERR_CHAR_ARRAY_ONLY	Deve essere un array di tipo char	GetLastError

Constant	Descrizione	Usage
ERR_CHART_CANNOT_CHANGE	Impossibile modificare il simbolo e periodo del grafico	GetLastError
ERR_CHART_CANNOT_CREATE_TIMER	Impossibile creare il timer	GetLastError
ERR_CHART_CANNOT_OPEN	Errore di apertura del grafico	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	Errore durante l'aggiunta di un indicatore al grafico	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	Errore durante l'eliminazione di un indicatore dal grafico	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	Indicatore non trovato sul grafico specificato	GetLastError
ERR_CHART_NAVIGATE_FAILED	Errore di navigazione attraver	GetLastError

Constant	Descrizione	Usage
	so il grafico	
ERR_CHART_NO_EXPERT	No Expert Advisor in the chart that could handle the event	GetLastError
ERR_CHART_NO_REPLY	Il grafico non risponde	GetLastError
ERR_CHART_NOT_FOUND	Grafico non trovato	GetLastError
ERR_CHART_SCREENSHOT_FAILED	Errore durante la creazione di uno screenshot	GetLastError
ERR_CHART_TEMPLATE_FAILED	Errore durante l'applicazione del template	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	La sottofinestra contenente l'indicatore non è stata trovata	GetLastError
ERR_CHART_WRONG_ID	ID del Grafico errato	GetLastError

Constant	Descrizione	Usage
ERR_CHART_WRONG_PARAMETER	Valore dell'errore dei parametri per la funzione di lavoro con i charts	GetLastError
ERR_CHART_WRONG_PROPERTY	ID proprietà del grafico, sbagliato	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	ID errato della proprietà indicatore personalizzato	GetLastError
ERR_DIRECTORY_NOT_EXIST	Directory non esiste	GetLastError
ERR_DOUBLE_ARRAY_ONLY	Deve essere un array di tipo double	GetLastError
ERR_FILE_BINSTRINGSIZE	La grandezza della stringa deve essere specificata, in quanto il file viene aperto come binario	GetLastError

Constant	Descrizione	Usage
ERR_FILE_CACHEBUFFER_ERROR	Memoria insufficiente per la cache di lettura	GetLastError
ERR_FILE_CANNOT_REWRITE	Il file non può essere riscritto	GetLastError
ERR_FILE_IS_DIRECTORY	Questo non è un file, questa è una directory	GetLastError
ERR_FILE_ISNOT_DIRECTORY	Questo è un file, non una directory	GetLastError
ERR_FILE_NOT_EXIST	Il file non esiste	GetLastError
ERR_FILE_NOTBIN	Il file deve essere aperto come un file binario	GetLastError
ERR_FILE_NOTCSV	Il file deve essere aperto come CSV	GetLastError
ERR_FILE_NOTTOREAD	Il file deve essere aperto per la lettura	GetLastError

Constant	Descrizione	Usage
ERR_FILE_NOTTOWRITE	Il file deve essere aperto per la scrittura	GetLastError
ERR_FILE_NOTTXT	Il file deve essere aperto come un testo	GetLastError
ERR_FILE_NOTTXTORCSV	Il file deve essere aperto come un testo o CSV	GetLastError
ERR_FILE_READERROR	Errore lettura file	GetLastError
ERR_FILE_WRITEERROR	Impossibile scrivere una risorsa in un file	GetLastError
ERR_FLOAT_ARRAY_ONLY	Deve essere un array di tipo float	GetLastError
ERR_FTP_SEND_FAILED	Invio file tramite ftp fallito	GetLastError
ERR_FUNCTION_NOT_ALLOWED	La funzione di sistema non è autorizz	GetLastError

Constant	Descrizione	Usage
	ata a chiamare	
ERR_GLOBALVARIABLE_EXISTS	La variabile globale del terminale client con lo stesso nome esiste già	GetLastError
ERR_GLOBALVARIABLE_NOT_FOUND	La variabile globale del terminale cliente non è stata trovata	GetLastError
ERR_HISTORY_NOT_FOUND	Storico richiesto non trovato	GetLastError
ERR_HISTORY_WRONG_PROPERTY	ID errato della proprietà dello storico	GetLastError
ERR_INCOMPATIBLE_ARRAYS	Copia di array incompatibili. Array di stringhe possono essere copiati solo per array di stringhe, ed un	GetLastError

Constant	Descrizione	Usage
	array numerico - solo in un array numerico	
ERR_INCOMPATIBLE_FILE	Un file di testo deve essere per gli array di stringhe, per altri array - binario	GetLastError
ERR_INDICATOR_CANNOT_ADD	Errore durante l'applicazione di un indicatore al grafico	GetLastError
ERR_INDICATOR_CANNOT_APPLY	L'indicatore non può essere applicato ad un altro indicatore	GetLastError
ERR_INDICATOR_CANNOT_CREATE	L'indicatore non può essere creato	GetLastError
ERR_INDICATOR_CUSTOM_NAME	Il primo parametro dell'array	GetLastError

Constant	Descrizione	Usage
	deve essere il nome dell'indicatore personalizzato	
ERR_INDICATOR_DATA_NOT_FOUND	Dati richiesti non trovati	GetLastError
ERR_INDICATOR_NO_MEMORY	Memoria insufficiente per aggiungere l'indicatore	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	Parametro di tipo non valido nell'array durante la creazione di un indicatore	GetLastError
ERR_INDICATOR_PARAMETERS_MISSING	Nessun parametro quando si crea un indicatore	GetLastError
ERR_INDICATOR_UNKNOWN_SYMBOL	Simbolo sconosciuto	GetLastError
ERR_INDICATOR_WRONG_HANDLE	Handle indicatore, errato	GetLastError

Constant	Descrizione	Usage
ERR_INDICATOR_WRONG_INDEX	Indice errato del buffer indicatore richiesto	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	Numero errato di parametri durante la creazione di un indicatore	GetLastError
ERR_INT_ARRAY_ONLY	Deve essere un array di tipo int	GetLastError
ERR_INTERNAL_ERROR	Errore interno imprevisto	GetLastError
ERR_INVALID_ARRAY	Array di un tipo errato, grandezza errata, o un oggetto danneggiato di un array dinamico	GetLastError
ERR_INVALID_DATETIME	Data e/o ora non validi	GetLastError
ERR_INVALID_FILEHANDLE	Un file con questo	GetLastError

Constant	Descrizione	Usage
	handle è stato chiuso, o non è stato proprio aperto	
ERR_INVALID_PARAMETER	Parametro errato quando si chiama la funzione di sistema	GetLastError
ERR_INVALID_POINTER	Puntatore errato	GetLastError
ERR_INVALID_POINTER_TYPE	Tipo errato di puntatore	GetLastError
ERR_LONG_ARRAY_ONLY	Deve essere un array di tipo long	GetLastError
ERR_MAIL_SEND_FAILED	Invio email fallito	GetLastError
ERR_MARKET_LASTTIME_UNKNOWN	L'orario dell'ultimo tick non è noto (nessun ticks)	GetLastError
ERR_MARKET_NOT_SELECTED	Il simbolo non è selezionato in	GetLastError

Constant	Descrizione	Usage
	Market Watch	
ERR_MARKET_SELECT_ERROR	Errore durante l'aggiunta o l'eliminazione di un simbolo in Market Watch	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	Simbolo sconosciuto	GetLastError
ERR_MARKET_WRONG_PROPERTY	Identificatore errato di una proprietà del simbolo	GetLastError
ERR_MQL5_WRONG_PROPERTY	Identificatore errato della proprietà del programma	GetLastError
ERR_NO_STRING_DATE	Nessuna data nella stringa	GetLastError
ERR_NOT_ENOUGH_MEMORY	Memoria insufficiente per eseguire la funzione del sistema	GetLastError

Constant	Descrizione	Usage
ERR_NOTIFICATION_SEND_FAILED	Fallimento nell'invio di una notifica	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	Invio troppo frequente delle notifiche	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	Parametro non valido per l'invio di una notifica - una stringa vuota o NULL è stato passato alla funzione SendNotification()	GetLastError
ERR_NOTIFICATION_WRONG_SETTINGS	Impostazioni errate di notifiche nel terminale (ID non è specificato o l'autorizzazione non è impostata)	GetLastError
ERR_NOTINITIALIZED_STRING	Stringa non	GetLastError

Constant	Descrizione	Usage
	inizializzata	
ERR_NUMBER_ARRAYS_ONLY	Deve essere un array numerico	GetLastError
ERR_OBJECT_ERROR	Errore di lavoro con un oggetto grafico	GetLastError
ERR_OBJECT_GETDATE_FAILED	Impossibile ottenere data corrispondente al valore	GetLastError
ERR_OBJECT_GETVALUE_FAILED	Impossibile ottenere il valore corrispondente alla data	GetLastError
ERR_OBJECT_NOT_FOUND	L' oggetto grafico non è stato trovato	GetLastError
ERR_OBJECT_WRONG_PROPERTY	ID errato di una proprietà oggetto grafico	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	Deve essere un array mono-	GetLastError

Constant	Descrizione	Usage
	dimensionale	
ERR_OPENCL_BUFFER_CREATE	Impossibile creare un buffer OpenCL	GetLastError
ERR_OPENCL_CONTEXT_CREATE	Errore durante la creazione del contesto OpenCL	GetLastError
ERR_OPENCL_EXECUTE	Errore programma OpenCL: errore di runtime	GetLastError
ERR_OPENCL_INTERNAL	Un errore interno si è verificato quando si eseguiva OpenCL	GetLastError
ERR_OPENCL_INVALID_HANDLE	Non valido handle OpenCL	GetLastError
ERR_OPENCL_KERNEL_CREATE	Errore durante la creazione di un OpenCL kernel	GetLastError

Constant	Descrizione	Usage
ERR_OPENCL_NOT_SUPPORTED	Funzioni OpenCL non sono supportate su questo computer	GetLastError
ERR_OPENCL_PROGRAM_CREATE	Si è verificato un errore durante la compilazione di un programma OpenCL	GetLastError
ERR_OPENCL_QUEUE_CREATE	Impossibile creare una coda di esecuzione in OpenCL	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	Errore avvenuto durante impostazione dei parametri per il kernel OpenCL	GetLastError
ERR_OPENCL_TOO_LONG_KERNEL_NAME	Nome kernel troppo lungo (OpenCL kernel)	GetLastError

Constant	Descrizione	Usage
ERR_OPENCL_WRONG_BUFFER_OFFSET	Offset non valido nel buffer OpenCL	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	Dimensione non valida del buffer OpenCL	GetLastError
ERR_PLAY_SOUND_FAILED	Riproduzione del suono fallita	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	I nomi delle risorse dinamiche e statiche, corrispondono	GetLastError
ERR_RESOURCE_NAME_IS_TOO_LONG	Il nome della risorsa supera i 63 caratteri	GetLastError
ERR_RESOURCE_NOT_FOUND	Risorse con questo nome non sono state trovate in EX5	GetLastError
ERR_RESOURCE_UNSUPPORTED_TYPE	Tipo di risorsa non supportata o la sua	GetLastError

Constant	Descrizione	Usage
	grandezza superai 16 Mb	
ERR_SERIES_ARRAY	Timeseries non possono essere utilizzate	GetLastError
ERR_SHORT_ARRAY_ONLY	Deve essere un array di tipo corto	GetLastError
ERR_SMALL_ARRAY	Array troppo piccolo, la posizione di partenza è fuori dell'array	GetLastError
ERR_SMALL_ASSERIES_ARRAY	L'array ricevente viene dichiarato come AS_SERIES, ed è di grandezza insufficiente	GetLastError
ERR_STRING_OUT_OF_MEMORY	Memoria insufficiente per la stringa	GetLastError
ERR_STRING_RESIZE_ERROR	Memoria insufficiente per il	GetLastError

Constant	Descrizione	Usage
	trasferimento di stringa	
ERR_STRING_SMALL_LEN	La lunghezza della stringa è inferiore al previsto	GetLastError
ERR_STRING_TIME_ERROR	Errore di conversione da stringa a data	GetLastError
ERR_STRING_TOO_BIGNUMBER	Numero troppo largo, più di ULONG_MAX	GetLastError
ERR_STRING_UNKNOWNTYPE	Dati di tipo sconosciuto durante la conversione in una stringa	GetLastError
ERR_STRING_ZEROADDED	0 aggiunto alla fine della stringa, una operazione inutile	GetLastError
ERR_STRINGPOS_OUTOFRANGE	Posizione al di fuori	GetLastError

Constant	Descrizione	Usage
	della stringa	
ERR_STRUCT_WITHOBJECTS_ORCLASS	La struttura contiene oggetti di stringhe e/o array dinamici e/o la struttura di tali oggetti e/o classi	GetLastError
ERR_SUCCESS	L'operazione è stata completata con successo	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	Identificatore errato della proprietà del terminale	GetLastError
ERR_TOO_LONG_FILENAME	Nome del file troppo lungo	GetLastError
ERR_TOO_MANY_FILES	Più di 64 file non possono essere aperti contemporaneamente	GetLastError

Constant	Descrizione	Usage
ERR_TOO_MANY_FORMATTERS	Quantità di specificatori di formato superiore ai parametri	GetLastError
ERR_TOO_MANY_PARAMETERS	Quantità dei parametri più degli specificatori di formato	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	Affare non trovato	GetLastError
ERR_TRADE_DISABLED	Trading da Expert Advisors vietato	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	Ordine non trovato	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	Posizione non trovata	GetLastError
ERR_TRADE_SEND_FAILED	Richiesta di trade non riuscita	GetLastError
ERR_TRADE_WRONG_PROPERTY	Proprietà ID del trade, sbagliata	GetLastError
ERR_USER_ERROR_FIRST	Errori definiti	GetLastError

Constant	Descrizione	Usage
	dall'utente iniziano con questo codice	
ERR_WEBREQUEST_CONNECT_FAILED	Failed to connect to specified URL	GetLastError
ERR_WEBREQUEST_INVALID_ADDRESS	Invalid URL	GetLastError
ERR_WEBREQUEST_REQUEST_FAILED	HTTP request failed	GetLastError
ERR_WEBREQUEST_TIMEOUT	Timeout exceeded	GetLastError
ERR_WRONG_DIRECTORYNAME	Nome directory errata	GetLastError
ERR_WRONG_FILEHANDLE	File handle errato	GetLastError
ERR_WRONG_FILENAME	Nome di file non validi	GetLastError
ERR_WRONG_FORMATSTRING	Formato stringa non valido	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	Parametri non validi nella chiamata interna della funzione del	GetLastError

Constant	Descrizione	Usage
	terminale client	
ERR_WRONG_STRING_DATE	Wrong date in the string	GetLastError
ERR_WRONG_STRING_OBJECT	Oggetto della stringa, dannegiato	GetLastError
ERR_WRONG_STRING_PARAMETER	Parametro danneggiato di tipo stringa	GetLastError
ERR_WRONG_STRING_TIME	Orario sbagliato nella stringa	GetLastError
ERR_ZEROSIZE_ARRAY	Un array di lunghezza zero	GetLastError
FILE_ACCESS_DATE	Data dell'ultimo accesso al file	FileGetInteger
FILE_ANSI	Stringhe di tipo ANSI (uno dei simboli byte). Flag viene usato in FileOpen()	FileOpen
FILE_BIN	Modalità binaria	FileOpen

Constant	Descrizione	Usage
	lettura/scrittura (senza conversione stringa a stringa). Flag viene usato in FileOpen()	
FILE_COMMON	Il percorso del file nella cartella comune di tutti i terminali client \Terminal\Common\Files. Il flag viene usato nelle funzioni FileOpen(), FileCopy(), FileMove() e FileIsExist().	FileOpen, FileCopy, FileMove, FileIsExist
FILE_CREATE_DATE	Data di creazione	FileGetInteger
FILE_CSV	CSV file (tutti i suoi elementi vengono converti	FileOpen

Constant	Descrizione	Usage
	ti in stringhe di tipo appropriato, unicode o ANSI, e separati da separatore). Flag viene usato in FileOpen()	
FILE_END	Prende la fine del segno del file	FileGetInteger
FILE_EXISTS	Controllare l'esistenza	FileGetInteger
FILE_IS_ANSI	Il file viene aperto come ANSI (vediFILE_ANSI)	FileGetInteger
FILE_IS_BINARY	Il file viene aperto come un file binario (vedi FILE_BIN)	FileGetInteger
FILE_IS_COMMON	Il file viene aperto in una	FileGetInteger

Constant	Descrizione	Usage
	cartella condivisa di tutti i terminali (vedi FILE_COMMON)	
FILE_IS_CSV	Il file viene aperto come CSV (vedi FILE_CSV)	FileGetInteger
FILE_IS_READABLE	Il file aperto è leggibile (vedi File_read)	FileGetInteger
FILE_IS_TEXT	Il file viene aperto come file di testo (vedi FILE_TXT)	FileGetInteger
FILE_IS_WRITABLE	Il file aperto è scrivibile (vedi File_write)	FileGetInteger
FILE_LINE_END	Prendi la fine del segno linea	FileGetInteger
FILE_MODIFY_DATE	Data dell'ultim	FileGetInteger

Constant	Descrizione	Usage
	a modifica	
FILE_POSITION	Posizione di un puntatore nel file	FileGetInteger
FILE_READ	Il file viene aperto per la lettura. Flag viene usato in FileOpen(). Quando si apre un file, la specifica di file FILE_WRITE e/o FILE_READ è necessaria.	FileOpen
FILE_REWRITE	Possibilità, per la riscrittura del file utilizzando le funzioni di FileCopy() e FileMove(). Il file deve esistere o deve essere aperto	FileCopy, FileMove

Constant	Descrizione	Usage
	in scrittura, altrimenti il file non viene aperto.	
FILE_SHARE_READ	L'accesso condiviso per la lettura da diversi programmi. Il flag viene usato in FileOpen(), ma non sostituisce la necessità di indicare FILE_WRITE e/o il flag FILE_READ all'apertura di un file.	FileOpen
FILE_SHARE_WRITE	L'accesso condiviso per la scrittura da diversi programmi. Il flag	FileOpen

Constant	Descrizione	Usage
	viene usato in FileOpen(), ma non sostituisce la necessità di indicare FILE_WRITE e/o il flag FILE_READ all'apertura di un file.	
FILE_SIZE	Dimensione del file in byte	FileGetInteger
FILE_TXT	File di testo semplice (lo stesso di file csv, ma senza prendere in considerazione i separatori). Flag viene usato in FileOpen()	FileOpen
FILE_UNICODE	Stringhe di tipo UNICODE (due simboli di byte).	FileOpen

Constant	Descrizione	Usage
	Flag viene usato in FileOpen()	
FILE_WRITE	Il file è aperto in scrittura. Flag viene usato in FileOpen(). Quando si apre un file, la specifica di file FILE_WRITE e/o FILE_READ è necessaria.	FileOpen
FLT_DIG	Numero di cifre decimali significative per il tipo float	Costanti di tipo numerico
FLT_EPSILON	Valore minimo, che soddisfa la condizione: 1.0+DBL_EPSILON != 1.0 (per il tipo float)	Costanti di tipo numerico

Constant	Descrizione	Usage
FLT_MANT_DIG	Conteggio Bits nella mantissa per il tipo float	Costanti di tipo numerico
FLT_MAX	Valore massimo, che può essere rappresentato dal tipo float	Costanti di tipo numerico
FLT_MAX_10_EXP	Massimo valore decimale del grado dell'esponente per il tipo float	Costanti di tipo numerico
FLT_MAX_EXP	Valore binario massimo dell' esponente del grado dell'esponente per il tipo float	Costanti di tipo numerico
FLT_MIN	Minimo valore positivo, che può essere rappresentato dal	Costanti di tipo numerico

Constant	Descrizione	Usage
	tipo float	
FLT_MIN_10_EXP	Valore decimale minimo del grado dell'esponente per il tipo float	Costanti di tipo numerico
FLT_MIN_EXP	Valore binario minimo del grado dell' esponente per il tipo float	Costanti di tipo numerico
FRIDAY	Venerdì	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
GANN_DOWN_TREND	Linea corrispondente alla tendenza al ribasso	ObjectSetInteger, ObjectGetInteger
GANN_UP_TREND	Linea corrispondente alla linea di trend rialzista	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	Linea Jaw	Linee Indicatori
GATORLIPS_LINE	Linea Lips	Linee Indicatori

Constant	Descrizione	Usage
GATORTEETH_LINE	Linea Teeth	Linee Indicatori
IDABORT	Bottone "Interruzione" è stato premuto	MessageBox
IDCANCEL	Bottone "Cancel" è stato premuto	MessageBox
IDCONTINUE	Bottone "Continua" è stato premuto	MessageBox
IDIGNORE	Bottone "Ignora" è stato premuto	MessageBox
IDNO	Bottone "No" è stato premuto	MessageBox
IDOK	Bottone "OK" è stato premuto	MessageBox
IDRETRY	Bottone "Riprova" è stato premuto	MessageBox
IDTRYAGAIN	Bottone "Prova Ancora" è stato premuto	MessageBox
IDYES	Bottone "Sì" è stato premuto	MessageBox

Constant	Descrizione	Usage
IND_AC	Accelerator Oscillator	IndicatorCreate, IndicatorParameters
IND_AD	Accumulation/Distribution	IndicatorCreate, IndicatorParameters
IND_ADX	Average Directional Index	IndicatorCreate, IndicatorParameters
IND_ADXW	ADX by Welles Wilder	IndicatorCreate, IndicatorParameters
IND_ALLIGATOR	Alligator	IndicatorCreate, IndicatorParameters
IND_AMA	Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_AO	Awesome Oscillator	IndicatorCreate, IndicatorParameters
IND_ATR	Average True Range	IndicatorCreate, IndicatorParameters
IND_BANDS	Bollinger Bands®	IndicatorCreate, IndicatorParameters
IND_BEARS	Bears Power	IndicatorCreate, IndicatorParameters
IND_BULLS	Bulls Power	IndicatorCreate, IndicatorParameters
IND_BWMFI	Market Facilitation Index	IndicatorCreate, IndicatorParameters
IND_CCI	Commodity Channel Index	IndicatorCreate, IndicatorParameters

Constant	Descrizione	Usage
IND_CHAIKIN	Chaikin Oscillator	IndicatorCreate, IndicatorParameters
IND_CUSTOM	Custom indicator	IndicatorCreate, IndicatorParameters
IND_DEMA	Double Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_DEMARKER	DeMarker	IndicatorCreate, IndicatorParameters
IND_ENVELOPES	Envelopes	IndicatorCreate, IndicatorParameters
IND_FORCE	Force Index	IndicatorCreate, IndicatorParameters
IND_FRACTALS	Fractals	IndicatorCreate, IndicatorParameters
IND_FRAMA	Fractal Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_GATOR	Gator Oscillator	IndicatorCreate, IndicatorParameters
IND_ICHIMOKU	Ichimoku Kinko Hyo	IndicatorCreate, IndicatorParameters
IND_MA	Moving Average	IndicatorCreate, IndicatorParameters
IND_MACD	MACD	IndicatorCreate, IndicatorParameters
IND_MFI	Money Flow Index	IndicatorCreate, IndicatorParameters
IND_MOMENTUM	Momentum	IndicatorCreate, IndicatorParameters
IND_OBV	On Balance Volume	IndicatorCreate, IndicatorParameters

Constant	Descrizione	Usage
IND_OSMA	OsMA	IndicatorCreate, IndicatorParameters
IND_RSI	Relative Strength Index	IndicatorCreate, IndicatorParameters
IND_RVI	Relative Vigor Index	IndicatorCreate, IndicatorParameters
IND_SAR	Parabolic SAR	IndicatorCreate, IndicatorParameters
IND_STDDEV	Standard Deviation	IndicatorCreate, IndicatorParameters
IND_STOCHASTIC	Stochastic Oscillator	IndicatorCreate, IndicatorParameters
IND_TEMA	Triple Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_TRIX	Triple Exponential Moving Averages Oscillator	IndicatorCreate, IndicatorParameters
IND_VIDYA	Variable Index Dynamic Average	IndicatorCreate, IndicatorParameters
IND_VOLUMES	Volumes	IndicatorCreate, IndicatorParameters
IND_WPR	Williams' Percent Range	IndicatorCreate, IndicatorParameters
INDICATOR_CALCULATIONS	Buffer ausiliari per calcoli	SetIndexBuffer

Constant	Descrizione	Usage
	intermedi	
INDICATOR_COLOR_INDEX	Color	SetIndexBuffer
INDICATOR_DATA	Dati per disegnare	SetIndexBuffer
INDICATOR_DIGITS	Precisione del disegno di valori degli indicatori	IndicatorSetInteger
INDICATOR_HEIGHT	Altezza fissa della finestra dell'indicatore (il comando del preprocessore #property indicator_height)	IndicatorSetInteger
INDICATOR_LEVELCOLOR	Colore della linea di livello	IndicatorSetInteger
INDICATOR_LEVELS	Numero di livelli nella finestra dell'indicatore	IndicatorSetInteger
INDICATOR_LEVELSTYLE	Stile della linea di livello	IndicatorSetInteger
INDICATOR_LEVELTEXT	Descrizione del Livello	IndicatorSetString

Constant	Descrizione	Usage
INDICATOR_LEVELVALUE	Valore del livello	IndicatorSetDouble
INDICATOR_LEVELWIDTH	Spessore della linea di livello	IndicatorSetInteger
INDICATOR_MAXIMUM	Massima della finestra dell'indicatore	IndicatorSetDouble
INDICATOR_MINIMUM	Minima della finestra dell'indicatore	IndicatorSetDouble
INDICATOR_SHORTNAME	Nome breve dell'indicatore	IndicatorSetString
INT_MAX	Valore massimo, che può essere rappresentato dal tipo int	Costanti di tipo numerico
INT_MIN	Valore minimo, che può essere rappresentato dal tipo int	Costanti di tipo numerico
INVALID_HANDLE	Incorrect handle	Altre costanti
IS_DEBUG_MODE	Flag dove un programma-mq5 opera in	Altre costanti

Constant	Descrizione	Usage
	modalità di debug	
IS_PROFILE_MODE	Flag dove un programma-mq5 opera in modalità di analisi	Altre costanti
KIJUNSEN_LINE	Linea Kijun-sen	Linee Indicatori
LICENSE_DEMO	Una versione di prova di un prodotto dal Market. Funziona solo nello strategy tester	MQLInfoInteger
LICENSE_FREE	Una versione gratuita ed illimitata	MQLInfoInteger
LICENSE_FULL	Una versione di licenza acquistata che permette almeno 5 attivazioni. Il venditore può	MQLInfoInteger

Constant	Descrizione	Usage
	incrementare il numero consentito di attivazioni	
LICENSE_TIME	Una versione con termini di licenza limitati	MQLInfoInteger
LONG_MAX	Valore massimo, che può essere rappresentato dal tipo long	Costanti di tipo numerico
LONG_MIN	Valore minimo, che può essere rappresentato dal tipo long	Costanti di tipo numerico
LOWER_BAND	Limite inferiore	Linee Indicatori
LOWER_HISTOGRAM	Istogramma inferiore	Linee Indicatori
LOWER_LINE	Linea inferiore	Linee Indicatori
M_1_PI	1/pi	Costanti Matematiche
M_2_PI	2/pi	Costanti Matematiche
M_2_SQRTPI	2/sqrt(pi)	Costanti Matematiche
M_E	e	Costanti Matematiche

Constant	Descrizione	Usage
M_LN10	ln(10)	Costanti Matematiche
M_LN2	ln(2)	Costanti Matematiche
M_LOG10E	log10(e)	Costanti Matematiche
M_LOG2E	log2(e)	Costanti Matematiche
M_PI	pi	Costanti Matematiche
M_PI_2	pi/2	Costanti Matematiche
M_PI_4	pi/4	Costanti Matematiche
M_SQRT1_2	1/sqrt(2)	Costanti Matematiche
M_SQRT2	sqrt(2)	Costanti Matematiche
MAIN_LINE	Linea principale	Linee Indicatori
MB_ABORTRETRYIGNORE	Finestra di messaggio contiene tre pulsanti: Interrompi, Riprova ed Ignora	MessageBox
MB_CANCELTRYCONTINUE	Finestra di messaggio contiene tre pulsanti: Annulla, Riprova, Continua	MessageBox
MB_DEFBUTTON1	Il primo bottone MB_DEFBUTTON1 - è di	MessageBox

Constant	Descrizione	Usage
	default, se gli altri pulsanti MB_DEFBUTTON2, MB_DEFBUTTON3 o MB_DEFBUTTON4 non sono specificati	
MB_DEFBUTTON2	Il secondo bottone è default	MessageBox
MB_DEFBUTTON3	Il terzo bottone è default	MessageBox
MB_DEFBUTTON4	Il quarto bottone è default	MessageBox
MB_ICONEXCLAMATION, MB_ICONWARNING	L'icona del segno di esclamazione/avvertimento	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	Il segno i cerchiata	MessageBox
MB_ICONQUESTION	L'icona del segno di punto interrogativo	MessageBox

Constant	Descrizione	Usage
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	L'icona del segno di STOP	MessageBox
MB_OK	La finestra di messaggio contiene un solo bottone: OK. Default	MessageBox
MB_OKCANCEL	Finestra di messaggio contiene due pulsanti: OK e Annulla	MessageBox
MB_RETRYCANCEL	Finestra di messaggio contiene due pulsanti: Riprova e Annulla	MessageBox
MB_YESNO	Finestra di messaggio contiene due pulsanti: Sì e No	MessageBox
MB_YESNOCANCEL	Finestra di messaggio	MessageBox

Constant	Descrizione	Usage
	contiene tre pulsanti: Sì, No e Annulla	
MINUSDI_LINE	Linea -DI	Linee Indicatori
MODE_EMA	Media esponenziale	Metodi di smussamento
MODE_LWMA	Media lineare-ponderata	Metodi di smussamento
MODE_SMA	Media semplice	Metodi di smussamento
MODE_SMMA	Media smussata	Metodi di smussamento
MONDAY	Lunedi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	Il flag, che indica la modalità di debug	MQLInfoInteger
MQL_DLLS_ALLOWED	Il permesso di utilizzare DLL per un determinato programma eseguito	MQLInfoInteger
MQL_FRAME_MODE	Il flag, che indica l'Expert	MQLInfoInteger

Constant	Descrizione	Usage
	Advisor opera in modalità raccolta frames dei risultati di ottimizzazione	
MQL_LICENSE_TYPE	Tipo di licenza del modulo EX5. La licenza si riferisce al modulo EX5, da cui viene effettuata una richiesta utilizzando MQLInfoInteger (MQL_LICENSE_TYPE).	MQLInfoInteger
MQL_MEMORY_LIMIT	Eventuale quantità massima di memoria dinamica per MQL5 programma in MB	MQLInfoInteger

Constant	Descrizione	Usage
MQL_MEMORY_USED	La grandezza della memoria utilizzata dal MQL5 programma in MB	MQLInfoInteger
MQL_OPTIMIZATION	Il flag, che indica il processo di ottimizzazione	MQLInfoInteger
MQL_PROFILER	Il flag, che indica il programma operante in modalità code profiling	MQLInfoInteger
MQL_PROGRAM_NAME	Nome del programma MQL5 eseguito	MQLInfoString
MQL_PROGRAM_PATH	Percorso per il programma eseguito dato	MQLInfoString
MQL_PROGRAM_TYPE	Tipo di programma MQL5	MQLInfoInteger

Constant	Descrizione	Usage
MQL_SIGNALS_ALLOWED	Il permesso di modificare i Signals per un determinato programma eseguito	MQLInfoInteger
MQL_TESTER	Il flag, che indica il processo di tester	MQLInfoInteger
MQL_TRADE_ALLOWED	L'autorizzazione al trade per un determinato programma eseguito	MQLInfoInteger
MQL_VISUAL_MODE	Il flag, che indica il processo di tester visuale	MQLInfoInteger
NULL	Zero per qualsiasi tipo	Altre costanti
OBJ_ALL_PERIODS	L'oggetto viene disegnato in tutti i timeframes	ObjectSetInteger, ObjectGetInteger
OBJ_ARROW	Freccia	Tipi di oggetti

Constant	Descrizione	Usage
OBJ_ARROW_BUY	Segno di Buy	Tipi di oggetti
OBJ_ARROW_CHECK	Segno di Visto	Tipi di oggetti
OBJ_ARROW_DOWN	Freccia giu	Tipi di oggetti
OBJ_ARROW_LEFT_PRICE	Etichetta Prezzo a Sinistra	Tipi di oggetti
OBJ_ARROW_RIGHT_PRICE	Etichetta Prezzo a Destra	Tipi di oggetti
OBJ_ARROW_SELL	Segno di Sell	Tipi di oggetti
OBJ_ARROW_STOP	Segno di Stop	Tipi di oggetti
OBJ_ARROW_THUMB_DOWN	Pollice giu	Tipi di oggetti
OBJ_ARROW_THUMB_UP	Pollice su	Tipi di oggetti
OBJ_ARROW_UP	Freccia su	Tipi di oggetti
OBJ_ARROWED_LINE	Linea con freccia	Tipi di oggetti
OBJ_BITMAP	Bitmap	Tipi di oggetti
OBJ_BITMAP_LABEL	Etichetta Bitmap	Tipi di oggetti
OBJ_BUTTON	Bottone	Tipi di oggetti
OBJ_CHANNEL	Canale Equidistante	Tipi di oggetti
OBJ_CHART	Grafico	Tipi di oggetti
OBJ_CYCLES	Linee Cicliche	Tipi di oggetti
OBJ_EDIT	Modifica	Tipi di oggetti

Constant	Descrizione	Usage
OBJ_ELLIOTWAVE3	L'onda correttiva Elliott	Tipi di oggetti
OBJ_ELLIOTWAVE5	L'onda motiva Elliott	Tipi di oggetti
OBJ_ELLIPSE	Ellisse	Tipi di oggetti
OBJ_EVENT	L'oggetto "Evento" corrispondente ad un evento nel calendario economico	Tipi di oggetti
OBJ_EXPANSION	L'espansione Fibonacci	Tipi di oggetti
OBJ_FIBO	Ritracciamento di Fibonacci	Tipi di oggetti
OBJ_FIBOARC	L' arco Fibonacci	Tipi di oggetti
OBJ_FIBOCHANNEL	Il canale Fibonacci	Tipi di oggetti
OBJ_FIBOFAN	Il ventaglio Fibonacci	Tipi di oggetti
OBJ_FIBOTIMES	Le Fibonacci Time Zones	Tipi di oggetti

Constant	Descrizione	Usage
OBJ_GANNFAN	In ventaglio Gann	Tipi di oggetti
OBJ_GANNGRID	La griglia Gann	Tipi di oggetti
OBJ_GANNLINE	La Linea Gann	Tipi di oggetti
OBJ_HLINE	Linea orizzontale	Tipi di oggetti
OBJ_LABEL	Etichetta	Tipi di oggetti
OBJ_NO_PERIODS	L'oggetto non viene disegnato in tutti i timeframes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_D1	L'oggetto viene disegnato nel grafico giornaliero	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H1	L'oggetto viene disegnato nel grafico 1-ora	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	L'oggetto viene disegnato nel grafico 12-ore	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	L'oggetto viene	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	disegnato nel grafico 2-ore	
OBJ_PERIOD_H3	L'oggetto viene disegnato nel grafico 3-ore	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H4	L'oggetto viene disegnato nel grafico 4-ore	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H6	L'oggetto viene disegnato nel grafico 6-ore	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H8	L'oggetto viene disegnato nel grafico 8-ore	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M1	L'oggetto viene disegnato nel grafico 1-minuto	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	L'oggetto viene disegnato nel grafico 10-minuti	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
OBJ_PERIOD_M12	L'oggetto viene disegnato nel grafico 12-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M15	L'oggetto viene disegnato nel grafico 2-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M2	L'oggetto viene disegnato nel grafico 2-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M20	L'oggetto viene disegnato nel grafico 20-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M3	L'oggetto viene disegnato nel grafico 3-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	L'oggetto viene disegnato nel grafico 30-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	L'oggetto viene disegnato nel	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	grafico 4-minuti	
OBJ_PERIOD_M5	L'oggetto viene disegnato nel grafico 5-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M6	L'oggetto viene disegnato nel grafico 6-minuti	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_MN1	L'oggetto viene disegnato nel grafico mensile	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_W1	L'oggetto viene disegnato nel grafico settimanale	ObjectSetInteger, ObjectGetInteger
OBJ_PITCHFORK	Andrews' Pitchfork	Tipi di oggetti
OBJ_RECTANGLE	Rettangolo	Tipi di oggetti
OBJ_RECTANGLE_LABEL	L'oggetto "Etichetta Rettangolo" per la creazione e la progettazione	Tipi di oggetti

Constant	Descrizione	Usage
	dell'interfaccia grafica personalizzata.	
OBJ_REGRESSION	Canale Regressione Lineare	Tipi di oggetti
OBJ_STDDEVCHANNEL	Canale Deviazione Standard	Tipi di oggetti
OBJ_TEXT	Testo	Tipi di oggetti
OBJ_TREND	Trend Line	Tipi di oggetti
OBJ_TRENDBYANGLE	Trend Line per Angolo	Tipi di oggetti
OBJ_TRIANGLE	Triangolo	Tipi di oggetti
OBJ_VLINE	Linea verticale	Tipi di oggetti
OBJPROP_ALIGN	Allineamento orizzontale del testo nell'oggetto "Edit" (OBJ_EDIT)	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	Posizione del punto di ancoraggio di un oggetto grafico	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANGLE	Angolo. Per gli	ObjectSetDouble, ObjectGetDouble

Constant	Descrizione	Usage
	oggetti con nessun angolo specificato, creato da un programma, il valore è pari a EMPTY_VALUE	
OBJPROP_ARROWCODE	Codice per l'oggetto Freccia	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	Oggetto sullo sfondo	ObjectSetInteger, ObjectGetInteger
OBJPROP_BGCOLOR	Il colore di sfondo per OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	ObjectSetInteger, ObjectGetInteger
OBJPROP_BMPFILE	Il nome del file-BMPper Bitmap Label. Vedi anche Risorse	ObjectSetString, ObjectGetString
OBJPROP_BORDER_COLOR	Colore del bordo per gli oggetti	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	OBJ_EDIT e OBJ_BUTTON	
OBJPROP_BORDER_TYPE	Tipo di bordo per l'oggetto "label Rectangle"	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	ID dell'oggetto "Chart" (OBJ_CHART). Esso permette di lavorare con le proprietà di questo oggetto, come con un chart normale utilizzando le funzioni descritte in Operazioni Chart, ma ci sono delle eccezioni.	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_SCALE	La scala per l'oggetto Chart	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
OBJPROP_COLOR	Color	ObjectSetInteger, ObjectGetInteger
OBJPROP_CORNER	L'angolo del chart per collegare un oggetto grafico	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	Orario di creazione di oggetti	ObjectSetInteger, ObjectGetInteger
OBJPROP_DATE_SCALE	Visualizzazione della scala temporale per l'oggetto Chart	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	Livello di Elliott Wave Marcatura	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEVIATION	Deviazione per il Canale di Deviazione Standard	ObjectSetDouble, ObjectGetDouble
OBJPROP_DIRECTION	Trend dell'oggetto Gann	ObjectSetInteger, ObjectGetInteger
OBJPROP_DRAWLINES	Visualizzazione righe per creare la Elliott Wave	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
OBJPROP_ELLIPSE	Mostra l'ellisse completa dell'oggetto Arco di Fibonacci (OBJ_FIBOARC)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FILL	Riempire un oggetto con il colore (per OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FONT	Font	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	Dimensione carattere	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	Proibire mostrando il nome di un oggetto grafico all'interno della	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	lista degli oggetti dal menu del terminale "Grafici" - "Oggetti" - "Lista degli Oggetti". Il valore true consente di nascondere un oggetto dalla lista. Per impostazione predefinita, true è impostato per gli oggetti che visualizzano gli eventi del calendario, la storia di trading ed agli oggetti creati da programmi MQL5.	

Constant	Descrizione	Usage
	Per vedere tali oggetti grafici e accedere alle loro proprietà, fare clic sul pulsante "Tutti" nella finestra "Lista degli oggetti".	
OBJPROP_LEVELCOLOR	Colore del livello-linea	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELS	Numero di livelli	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	Stile della linea-di-livello	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELTEXT	Descrizione del Livello	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	Valore del livello	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	Spessore della linea-di-livello	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	Nome oggetto	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	Timeframe per l'oggetto	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	del Chart	
OBJPROP_PRICE	Coordinate Prezzo	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	Visualizzazione della scala di prezzo per l'oggetto Chart	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY	Una linea verticale passa attraverso tutte le finestre di un grafico	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	Il raggio va a sinistra	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_RIGHT	Il raggio va a destra	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	Possibilità di modificare il testo nell' oggetto Edit	ObjectSetInteger, ObjectGetInteger
OBJPROP_SCALE	Scala (proprietà degli oggetti di Gann e	ObjectSetDouble, ObjectGetDouble

Constant	Descrizione	Usage
	Fibonacci Arcs)	
OBJPROP_SELECTABLE	Disponibilità oggetto	ObjectSetInteger, ObjectGetInteger
OBJPROP_SELECTED	L'oggetto è selezionato	ObjectSetInteger, ObjectGetInteger
OBJPROP_STATE	Status del Bottone (premuto/non premuto)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	Style	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	Simbolo per l'oggetto Chart	ObjectSetString, ObjectGetString
OBJPROP_TEXT	Descrizione dell'oggetto (il testo contenuto nell'oggetto)	ObjectSetString, ObjectGetString
OBJPROP_TIME	Coordinate temporali	ObjectSetInteger, ObjectGetInteger
OBJPROP_TIMEFRAMES	La visibilità di un oggetto al timeframes	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
OBJPROP_TOOLTIP	Il testo del tooltip. Se la proprietà non è impostata, allora viene visualizzato il tooltip generato automaticamente dal terminale. Un tooltip può essere disabilitato impostando il valore "\n" (line feed) ad esso	ObjectSetString, ObjectGetString
OBJPROP_TYPE	Tipo di oggetto	ObjectSetInteger, ObjectGetInteger
OBJPROP_WIDTH	Spessore della linea	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	La distanza in pixel lungo l'asse X dal angolo vincolante (see note)	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
OBJPROP_XOFFSET	La coordinata X dell'angolo superiore sinistro della area visibile rettangolare negli oggetti grafici "Label Bitmap" e "Bitmap" (OBJ_BITMAP_LABEL e OBJ_BITMAP). Il valore è impostato in pixel rispetto all'angolo superiore sinistro dell'immagine originale.	ObjectSetInteger, ObjectGetInteger
OBJPROP_XSIZE	Larghezza dell'oggetto lungo l'asse X in pixel. Specificato per	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	oggetti OBJ_LABEL (solo lettura), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_YDISTANCE	La distanza in pixel lungo l'asse Y dall'angolo vincolante (see note)	ObjectSetInteger, ObjectGetInteger
OBJPROP_YOFFSET	La coordinata Y dell'angolo superiore sinistro dell' area visibile rettangolare negli oggetti grafici "Label Bitmap" e	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	"Bitmap" (OBJ_BITMAP_LABEL e OBJ_BITMAP). Il valore è impostato in pixel rispetto all'angolo superiore sinistro dell'immagine originale.	
OBJPROP_YSIZE	L'altezza dell'oggetto lungo l'asse Y in pixel. Specificato per oggetti OBJ_LABEL (solo lettura), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
OBJPROP_ZORDER	La priorità di un oggetto grafico per la ricezione degli eventi cliccando su un grafico (CHART_EVENT_CLICK). Il valore zero predefinito viene impostato durante la creazione di un oggetto; la priorità può essere aumentata se necessario. Quando si applica uno degli oggetti su un altro, solo uno di essi con la priorità più alta riceverà	ObjectSetInteger, ObjectGetInteger

Constant	Descrizione	Usage
	l'evento CHARTEVENT_CLICK.	
ORDER_COMMENT	Commento all'ordine	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	Questa politica di filling significa che un ordine può essere riempito solo nella quantità specificata. Se la quantità necessaria di uno strumento finanziario non è attualmente disponibile sul mercato, l'ordine non verrà eseguito. Il volume richiesto può essere riempito con diverse offerte	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
	disponibili sul mercato al momento.	
ORDER_FILLING_IOC	Questa modalità significa che un trader si impegna ad eseguire un affare con il volume massimamente disponibile sul mercato entro quello indicata nell'ordine. Nel caso in cui il l'intero volume di un ordine non può essere riempito, verrà riempito il volume disponibile di esso, e il restante volume	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
	verrà annullato.	
ORDER_FILLING_RETURN	Questa policy viene utilizzata solo per ordini di mercato (ORDER_TYPE_BUY e ORDER_TYPE_SELL), ordini limit e stop limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT) e solo per i simboli con esecuzione Market o Exchange. In caso il riempim	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
	ento di un ordine a mercato o ordine limit con volume rimanente parziale non venga annullato, ma ulteriormente elaborato. Per l'attivazione degli ordini ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT, viene creato un corrispondente limit order ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT con l'esecuzione ORDER_	

Constant	Descrizione	Usage
	FILLING_RETURN.	
ORDER_MAGIC	ID di un Expert Advisor che ha piazzato l'ordine (progettato per garantire che ogni Expert Advisor colloca il proprio numero unico)	OrderGetInteger, HistoryOrderGetInteger
ORDER_POSITION_ID	Identificatore Posizione che è impostato in un ordine non appena viene eseguito. Ogni ordine eseguito risulta in un affare che apre o modifica una posizione già esistente. L'identifi	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
	catore di esattamente questa posizione viene impostato ad ordine eseguito, in questo momento.	
ORDER_PRICE_CURRENT	Il prezzo attuale del simbolo dell'ordine	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_OPEN	Prezzo specificato nell'ordine	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	Il prezzo Limit dell' ordine per l'ordine StopLimit	OrderGetDouble, HistoryOrderGetDouble
ORDER_SL	Valore Stop Loss	OrderGetDouble, HistoryOrderGetDouble
ORDER_STATE	Stato dell'ordine	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	Ordine annullato dal client	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
ORDER_STATE_EXPIRED	Ordine espirato	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	Ordine pienamente eseguito	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	Ordine parzialmente eseguito	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	Ordine accettato	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REJECTED	Ordine rigettato	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	L' ordine è stato registrato (piazzato al trading system)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_CANCEL	L'ordine è stato eliminato (eliminato dal trading system)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	L'ordine è stato modificato (cambio dei suoi parametri)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	Ordine controllato, ma non	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
	ancora accettato dal broker	
ORDER_SYMBOL	Smbolo dell'ordine	OrderGetString, HistoryOrderGetString
ORDER_TIME_DAY	Buono sino al corrente ordine d giorno di trade	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE	Orario di esecuzione o eliminazione dell' Ordine	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	Orario di esecuzione/eliminazion di ordini in millisecondi dal 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	Orario di espirazione dell'ordine	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	Buona fino a cancellazione ordine	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP	Setup dell'orario dell' Ordine	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
ORDER_TIME_SETUP_MSC	Lìorario di piazzamento dell'esecuzione di un ordine in millisecondi dal 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED	Buona fino a espirazione ordine	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	L'ordine sarà efficace fino a 23:59:59 del giorno specificato. Se questo orario è al di fuori di una sessione di trading, l'ordine scade nel più vicino orario di trading.	OrderGetInteger, HistoryOrderGetInteger
ORDER_TP	Valore Take Profit	OrderGetDouble, HistoryOrderGetDouble
ORDER_TYPE	Tipo di ordine	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
ORDER_TYPE_BUY	Ordine di mercato Buy	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	Ordine pendente Buy Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	Ordine pendente Buy Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	Dopo aver raggiunto il prezzo dell'ordine, un ordine pentende Buy Limit viene piazzato al prezzo StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_FILLING	Tipo di riempimento dell'ordine	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL	Ordine di mercato Sell	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	Ordine pendente Sell Limit	OrderGetInteger, HistoryOrderGetInteger

Constant	Descrizione	Usage
ORDER_TYPE_SELL_STOP	Ordine pendente Sell Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	Dopo aver raggiunto il prezzo dell'ordine, un ordine pendente Sell Limit viene piazzato al prezzo StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_TIME	Durata dell'ordine	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	Volume corrente dell'ordine	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	Volume iniziale dell'ordine	OrderGetDouble, HistoryOrderGetDouble
PERIOD_CURRENT	Timeframe corrente	Timeframes del Grafico
PERIOD_D1	1 giorno	Timeframes del Grafico
PERIOD_H1	1 ora	Timeframes del Grafico
PERIOD_H12	12 ore	Timeframes del Grafico
PERIOD_H2	2 ore	Timeframes del Grafico
PERIOD_H3	3 ore	Timeframes del Grafico
PERIOD_H4	4 ore	Timeframes del Grafico

Constant	Descrizione	Usage
PERIOD_H6	6 ore	Timeframes del Grafico
PERIOD_H8	8 ore	Timeframes del Grafico
PERIOD_M1	1 minuto	Timeframes del Grafico
PERIOD_M10	10 minuti	Timeframes del Grafico
PERIOD_M12	12 minuti	Timeframes del Grafico
PERIOD_M15	15 minuti	Timeframes del Grafico
PERIOD_M2	2 minuti	Timeframes del Grafico
PERIOD_M20	20 minuti	Timeframes del Grafico
PERIOD_M3	3 minuti	Timeframes del Grafico
PERIOD_M30	30 minuti	Timeframes del Grafico
PERIOD_M4	4 minuti	Timeframes del Grafico
PERIOD_M5	5 minuti	Timeframes del Grafico
PERIOD_M6	6 minuti	Timeframes del Grafico
PERIOD_MN1	1 mese	Timeframes del Grafico
PERIOD_W1	1 settimana	Timeframes del Grafico
PLOT_ARROW	Codice Arrow per DRAW_ARROW stile	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_ARROW_SHIFT	Spostamento verticale di frecce per stile DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_COLOR_INDEXES	Il numero	PlotIndexSetInteger, PlotIndexGetInteger

Constant	Descrizione	Usage
	di colori	
PLOT_DRAW_BEGIN	Numero di barre iniziali, senza disegno e valori nel DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_TYPE	Tipo di costruzione grafica	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_EMPTY_VALUE	Un valore vuoto per la stampa, per cui non esiste nessun disegno	PlotIndexSetDouble
PLOT_LABEL	Il nome della serie grafica dell'indicatore da visualizzare nel DataWindow. Quando si lavora con stili grafici complessi che richiedono diversi buffer	PlotIndexSetString

Constant	Descrizione	Usage
	indicatori per la visualizzazione, i nomi per ogni buffer possono essere specificati utilizzando ";" come separatore. Codice di esempio è mostrato in DRAW_CANDLES	
PLOT_LINE_COLOR	L'indice di un tampone contenente il colore di disegno	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_STYLE	Stile di Disegno della linea	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_WIDTH	Lo spessore della linea di disegno	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHIFT	Spostamento dell'indicatore tracciato	PlotIndexSetInteger, PlotIndexGetInteger

Constant	Descrizione	Usage
	lungo l'asse del tempo nelle bars	
PLOT_SHOW_DATA	Segno di visualizzazione dei valori di costruzione nel DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLUSDI_LINE	Linea +DI	Linee Indicatori
POINTER_AUTOMATIC	Puntatore di tutti gli oggetti creati automaticamente (non utilizzando new())	CheckPointer
POINTER_DYNAMIC	Puntatore dell'oggetto creato dall'operatore new()	CheckPointer
POINTER_INVALID	Puntatore errato	CheckPointer
POSITION_COMMENT	Commento della Posizione	PositionGetString
POSITION_COMMISSION	Commission	PositionGetDouble

Constant	Descrizione	Usage
POSITION_IDENTIFIER	L'identificatore di posizione è un numero unico che viene assegnato ad ogni posizione nuovamente aperta e non cambia durante l'intero periodo di vita della posizione. Il turnover della posizione non cambia il suo identificatore.	PositionGetInteger
POSITION_MAGIC	Numero magico di posizione (vedi ORDER_MAGIC)	PositionGetInteger
POSITION_PRICE_CURRENT	Prezzo corrente del simbolo della	PositionGetDouble

Constant	Descrizione	Usage
	posizione	
POSITION_PRICE_OPEN	Prezzo di apertura della posizione	PositionGetDouble
POSITION_PROFIT	Profitto corrente	PositionGetDouble
POSITION_SL	Livello di Stop Loss della posizione aperta	PositionGetDouble
POSITION_SWAP	Swap cumulativo	PositionGetDouble
POSITION_SYMBOL	Simbolo della posizione	PositionGetString
POSITION_TIME	Orario di apertura della posizione	PositionGetInteger
POSITION_TIME_MSC	Orario di apertura della posizione in millisecondi dal 01.01.1970	PositionGetInteger
POSITION_TIME_UPDATE	Orario di cambio della posizione in secondi	PositionGetInteger

Constant	Descrizione	Usage
	dal 01.01.1970	
POSITION_TIME_UPDATE_MSC	Orario di cambio della posizione in millisecondi dal 01.01.1970	PositionGetInteger
POSITION_TP	Livello di Take Profit della posizione aperta	PositionGetDouble
POSITION_TYPE	Tipo di posizione	PositionGetInteger
POSITION_TYPE_BUY	Buy	PositionGetInteger
POSITION_TYPE_SELL	Sell	PositionGetInteger
POSITION_VOLUME	Volume della posizione	PositionGetDouble
PRICE_CLOSE	Prezzo di chiusura	Costanti Prezzo
PRICE_HIGH	Il prezzo massimo per il periodo	Costanti Prezzo
PRICE_LOW	Il prezzo minimo per il periodo	Costanti Prezzo
PRICE_MEDIAN	Prezzo mediano	Costanti Prezzo

Constant	Descrizione	Usage
	, (high + low)/2	
PRICE_OPEN	Prezzo di apertura	Costanti Prezzo
PRICE_TYPICAL	Prezzo tipico, (high + low + close)/3	Costanti Prezzo
PRICE_WEIGHTED	Prezzo medio, (alto + basso + vicino + close)/4	Costanti Prezzo
PROGRAM_EXPERT	Expert	MQLInfoInteger
PROGRAM_INDICATOR	Indicatore	MQLInfoInteger
PROGRAM_SCRIPT	Script	MQLInfoInteger
REASON_ACCOUNT	Un altro account è stato attivato o si è verificata la riconnessione al trade server a causa di cambiamenti nelle impostazioni dell'account	UninitializeReason, OnDeinit
REASON_CHARTCHANGE	Il simbolo o un periodo	UninitializeReason, OnDeinit

Constant	Descrizione	Usage
	del grafico è stato cambiato	
REASON_CHARTCLOSE	Il grafico è stato chiuso	UninitializeReason, OnDeinit
REASON_CLOSE	Il terminale è stato chiuso	UninitializeReason, OnDeinit
REASON_INITFAILED	Questo valore indica che l'handler OnInit() ha restituito un valore diverso da zero	UninitializeReason, OnDeinit
REASON_PARAMETERS	I parametri di input sono stati modificati dall'utente	UninitializeReason, OnDeinit
REASON_PROGRAM	Expert Advisor terminata la loro operazione chiamando la funzione ExpertRemove()	UninitializeReason, OnDeinit

Constant	Descrizione	Usage
REASON_RECOMPILE	Il programma è stato ricompilato	UninitializeReason, OnDeinit
REASON_REMOVE	Il programma è stato eliminato dal grafico	UninitializeReason, OnDeinit
REASON_TEMPLATE	Un nuovo template è stato applicato	UninitializeReason, OnDeinit
SATURDAY	Sabato	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	Posizione attuale di un puntatore a file	FileSeek
SEEK_END	Fine file	FileSeek
SEEK_SET	Inizio file	FileSeek
SENKOUSPANA_LINE	Linea Senkou Span A	Linee Indicatori
SENKOUSPANB_LINE	Linea Senkou Span B	Linee Indicatori
SERIES_BARS_COUNT	Conteggio barre per il simbolo-periodo per il	SeriesInfoInteger

Constant	Descrizione	Usage
	momento attuale	
SERIES_FIRSTDATE	La prima vera data per il simbolo-periodo per il momento attuale	SeriesInfoInteger
SERIES_LASTBAR_DATE	Tempo di apertura dell'ultima barra del simbolo-periodo	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	La prima data nello storico del simbolo sul server, indipendentemente dal periodo di tempo	SeriesInfoInteger
SERIES_SYNCHRONIZED	Flag della data di sincronizzazione per il simbolo/periodo per il momento attuale	SeriesInfoInteger

Constant	Descrizione	Usage
SERIES_TERMINAL_FIRSTDATE	La prima data nello storico del simbolo nel terminale client, indipendentemente dal timeframe	SeriesInfoInteger
SHORT_MAX	Valore massimo, che può essere rappresentato dal tipo short	Costanti di tipo numerico
SHORT_MIN	Valore minimo, che può essere rappresentato dal tipo short	Costanti di tipo numerico
SIGNAL_BASE_AUTHOR_LOGIN	Login Autore	SignalBaseGetString
SIGNAL_BASE_BALANCE	Account: bilancio	SignalBaseGetDouble
SIGNAL_BASE_BROKER	Nome del Broker (società)	SignalBaseGetString
SIGNAL_BASE_BROKER_SERVER	Server del Broker	SignalBaseGetString

Constant	Descrizione	Usage
SIGNAL_BASE_CURRENCY	Signal base currency	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	Data di pubblicazione (la data di quando è diventato disponibile per la sottoscrizione)	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	Data di inizio monitoraggio	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	Account: equità	SignalBaseGetDouble
SIGNAL_BASE_GAIN	Account: guadagno	SignalBaseGetDouble
SIGNAL_BASE_ID	ID del Segnale	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	Leverage dell'account	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	Account: massimo drawdown	SignalBaseGetDouble
SIGNAL_BASE_NAME	Nome del segnale	SignalBaseGetString
SIGNAL_BASE_PIPS	Prifitti in pips	SignalBaseGetInteger
SIGNAL_BASE_PRICE	Prezzo di	SignalBaseGetDouble

Constant	Descrizione	Usage
	sottoscrizione al Segnale	
SIGNAL_BASE_RATING	Posizione nel rating	SignalBaseGetInteger
SIGNAL_BASE_ROI	Ritorno dell' Investimento(%)	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	Numero di sottoscritti	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	Tipo di account (0-reale, 1-demo, 2-contest)	SignalBaseGetInteger
SIGNAL_BASE_TRADES	Numero di trades	SignalBaseGetInteger
SIGNAL_INFO_CONFIRMATIONS_DISABLED	The flag enables synchronization without confirmation dialog	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_COPY_SLTP	Flag di Copia Stop Loss e Take Profit	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	Deposito in percentuale (%)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	Limite equità	SignalInfoGetDouble, SignalInfoSetDouble

Constant	Descrizione	Usage
SIGNAL_INFO_ID	Il Signal id	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	Nome del segnale	SignalInfoGetString
SIGNAL_INFO_SLIPPAGE	Lo slippage (utilizzato quando si posizionano ordini di mercato nella sincronizzazione di posizioni e la copia dei trades)	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_SUBSCRIPTION_ENABLED	"Copy trades by subscription" permission flag	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	Flag "Acconsento ai termini di uso del servizio Segnali"	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	Massimspercentuale di utiilizzo del	SignalInfoGetDouble, SignalInfoSetDouble

Constant	Descrizione	Usage
	deposito (%)	
SIGNAL_LINE	Linea di segnale	Linee Indicatori
STAT_BALANCE_DD	Drawdown massimo del bilancio in termini monetari. Nel processo di trading, un bilancio può avere numerosi drowdowns; qui viene preso il valore più grande	TesterStatistics
STAT_BALANCE_DD_RELATIVE	Drawdown bilancio in termini monetari che è stato registrato al momento del drowdown massimo del bilancio	TesterStatistics

Constant	Descrizione	Usage
	in percentuale (STAT_BALANCE_DDREL_PERCENT).	
STAT_BALANCE_DDREL_PERCENT	Drawdown bilancio massimo in percentuale. Nel processo di trading, un bilancio può avere numerosi drowdowns, per ciascuno dei quali viene calcolato il relativo valore drowdown in percentuale. Viene restituito il maggior valore	TesterStatistics
STAT_BALANCEDD_PERCENT	Drawdown bilancio come	TesterStatistics

Constant	Descrizione	Usage
	percentuale che è stata registrata al momento del drawdown massimo del bilancio in termini monetari (STAT_BALANCE_DD).	
STAT_BALANCEMIN	Valore minimo del bilancio	TesterStatistics
STAT_CONLOSSMAX	La perdita massima in una serie di trades perdenti. Il valore è minore o uguale a zero	TesterStatistics
STAT_CONLOSSMAX_TRADES	Il numero di transazioni che si sono formate STAT_CONLOSSMAX (perdita	TesterStatistics

Constant	Descrizione	Usage
	massima in una serie di trades perdenti)	
STAT_CONPROFITMAX	Massimo profitto in una serie di trades profittevoli. Il valore è maggiore o uguale a zero	TesterStatistics
STAT_CONPROFITMAX_TRADES	Il numero di trades che viene formato STAT_CONPROFITMAX (massimo profitto in una serie di trades profittevoli)	TesterStatistics
STAT_CUSTOM_ONTESTER	Il valore del criterio di ottimizzazione personalizzata calcolato, restituit	TesterStatistics

Constant	Descrizione	Usage
	o dalla funzione OnTester()	
STAT_DEALS	Il numero di affari	TesterStatistics
STAT_EQUITY_DD	Valore massimo dell'rquità in termini monetari. Nel processo di trading, numerosi drowdowns possono essere visualizzati sul sull'equità; qui viene preso il valore più grande.	TesterStatistics
STAT_EQUITY_DD_RELATIVE	Drowdown dell'equità in termini monetari che sono stati registrati al momento del drowdow	TesterStatistics

Constant	Descrizione	Usage
	n massimo dell'equità, in percentuale(STAT_EQUITY_DDREL_PERCENT).	
STAT_EQUITY_DDREL_PERCENT	Drawdown massomo dell'equità come percentuale. Nel processo di trading, un'equità può avere numerosi drawdowns, per ciascuno dei quali viene calcolato il valore drowdown relativo in percentuale. Viene restituito il maggior valore	TesterStatistics
STAT_EQUITYDD_PERCENT	Drawdown in	TesterStatistics

Constant	Descrizione	Usage
	percentuale che è stato registrato al momento del drowdown massimo dell'equità in termini monetari (STAT_EQUITY_DD).	
STAT_EQUITYMIN	Valore minimo dell'equità	TesterStatistics
STAT_EXPECTED_PAYOFF	Payoff atteso	TesterStatistics
STAT_GROSS_LOSS	Perdita totale, la somma di tutti i trades negativi. Il valore è minore o uguale a zero	TesterStatistics
STAT_GROSS_PROFIT	Profitto totale netto, la somma di tutti i trades profittevoli (positivi	TesterStatistics

Constant	Descrizione	Usage
	). Il valore è maggiore o uguale a zero	
STAT_INITIAL_DEPOSIT	Il valore del deposito iniziale	TesterStatistics
STAT_LONG_TRADES	Trades long	TesterStatistics
STAT_LOSS_TRADES	Trades perdenti	TesterStatistics
STAT_LOSSTRADES_AVGCON	Lunghezza media di una serie di trades perdenti	TesterStatistics
STAT_MAX_CONLOSS_TRADES	Il numero di trades nella più lunga serie di trades perdenti STAT_MAX_CONLOSSES	TesterStatistics
STAT_MAX_CONLOSSES	La perdita totale della più lunga serie di trades perdenti	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	Il numero di trades	TesterStatistics

Constant	Descrizione	Usage
	nella più lunga serie di trades profittevoli STAT_MAX_CONWINS	
STAT_MAX_CONWINS	Il profitto totale della più lunga serie di trades profittevoli	TesterStatistics
STAT_MAX_LOSSTRADE	Massima perdita - il valore più basso di tutti i trades perdenti. Il valore è minore o uguale a zero	TesterStatistics
STAT_MAX_PROFITTRADE	Massimo profitto - il valore più grande di tutti i trades profittevoli. Il valore è maggiore o uguale a zero	TesterStatistics

Constant	Descrizione	Usage
STAT_MIN_MARGINLEVEL	Valore minimo del livello di margine	TesterStatistics
STAT_PROFIT	Il profitto netto dopo il testing, la somma di STAT_GROSS_PROFIT e STAT_GROSS_LOSS (STAT_GROSS_LOSS è sempre inferiore o uguale a zero)	TesterStatistics
STAT_PROFIT_FACTOR	Fattore profitto, uguale al rapporto di STAT_GROSS_PROFIT / STAT_GROSS_LOSS. Se STAT_GROSS_LOSS=0, il fattore di profitto è pari a DBL_MAX	TesterStatistics

Constant	Descrizione	Usage
STAT_PROFIT_LONGTRADES	Trades long profittevoli	TesterStatistics
STAT_PROFIT_SHORTTRADES	Trades short profittevoli	TesterStatistics
STAT_PROFIT_TRADES	Trades profittevoli	TesterStatistics
STAT_PROFITTRADES_AVGCON	Lunghezza media di una serie di trades profittevoli	TesterStatistics
STAT_RECOVERY_FACTOR	Fattore di recupero, pari al rapporto di STAT_PROFIT / STAT_BALANCE_DD	TesterStatistics
STAT_SHARPE_RATIO	Indice di Sharpe	TesterStatistics
STAT_SHORT_TRADES	Trades short	TesterStatistics
STAT_TRADES	Il numero di trades	TesterStatistics
STAT_WITHDRAWAL	Il denaro prelevato da un conto	TesterStatistics

Constant	Descrizione	Usage
STO_CLOSECLOSE	Calculation is based on Close/Close prices	Costanti Prezzo
STO_LOWHIGH	Il calcolo si basa sui prezzi inferiore/superiore	Costanti Prezzo
STYLE_DASH	Linea spezzata	Stili di Disegno
STYLE_DASHDOT	Linea punteggiata-tratteggiata	Stili di Disegno
STYLE_DASHDOTDOT	Tratteggio - due punti	Stili di Disegno
STYLE_DOT	Linea tratteggiata	Stili di Disegno
STYLE_SOLID	Linea continua	Stili di Disegno
SUNDAY	Domenica	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	Ask - migliore offerta buy	SymbolInfoDouble
SYMBOL_ASKHIGH	Massimo Ask del giorno	SymbolInfoDouble
SYMBOL_ASKLOW	Minimo Ask del giorno	SymbolInfoDouble

Constant	Descrizione	Usage
SYMBOL_BANK	Feeder della corrente quotazione	SymbolInfoString
SYMBOL_BASIS	The underlying asset of a derivative	SymbolInfoString
SYMBOL_BID	Bid - migliore offerta sell	SymbolInfoDouble
SYMBOL_BIDHIGH	Massimo Bid del giorno	SymbolInfoDouble
SYMBOL_BIDLOW	Minimo Bid del giorno	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	Modalità CFD - calcolo del margine e del profitto e CFD	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	Modalità indici CFD - calcolo del margine e del profitto per indici CFD	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDLEVERAGE	Modalità CFD Leverag	SymbolInfoInteger

Constant	Descrizione	Usage
	e - calcolo del margine e del profitto per CFD al trading leverage	
SYMBOL_CALC_MODE_EXCH_FUTURES	Modalità Futures - calcolo del margine e del profitto per i contratti futures in uno stock exchange / borsa	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	Modalità FORTS Futures - calcolo del margine e del profitto per i contratti futures sui FORTS.	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_STOCKS	Modalità di Scambio - calcolo del margine e del profitto	SymbolInfoInteger

Constant	Descrizione	Usage
	per i titoli di trade in uno stock exchange / borsa	
SYMBOL_CALC_MODE_FOREX	Modalità Forex - calcolo del margine e del di profitto per Forex	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	Modalità Futures - calcolo del margine e del profitto per i futures	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - a symbol is used as a non-tradable asset on a trading account.	SymbolInfoInteger
SYMBOL_CURRENCY_BASE	Valuta base di un simbolo	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	Valuta di margine	SymbolInfoString

Constant	Descrizione	Usage
SYMBOL_CURRENCY_PROFIT	Valuta di profitto	SymbolInfoString
SYMBOL_DESCRIPTION	Descrizione del simbolo	SymbolInfoString
SYMBOL_DIGITS	Cifre dopo la virgola decimale	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	L'ordine è valido fino alla fine della giornata	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	L'ordine è valido durante un periodo di tempo illimitato, fino a quando non viene esplicitamente annullato	SymbolInfoInteger
SYMBOL_EXPIRATION_MODE	Flags di ordine consentito modalità di espirazione	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED	L'orario di espirazione viene specifica	SymbolInfoInteger

Constant	Descrizione	Usage
	to nell'ordine	
SYMBOL_EXPIRATION_SPECIFIED_DAY	L'orario di scadenza è specificato nell'ordine	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	Data di scadenza del commercio di trade (di solito usato per i futures)	SymbolInfoInteger
SYMBOL_FILLING_FOK	Questa regola significa che un affare può essere eseguito solo con il volume specificato. Se la quantità necessaria di uno strumento finanziario non è attualmente disponibile sul mercato	SymbolInfoInteger

Constant	Descrizione	Usage
	, l'ordine non verrà eseguito. Il volume richiesto può essere riempito con diverse offerte disponibili sul mercato al momento.	
SYMBOL_FILLING_IOC	In questo caso un trader si impegna ad eseguire un affare con il volume massimo disponibile sul mercato entro quello indicato nell'ordine. Nel caso in cui l'ordine non può essere riempito completamente,	SymbolInfoInteger

Constant	Descrizione	Usage
	il volume disponibile dell'ordine sarà riempito, e il volume rimanente verrà annullato. La possibilità di utilizzare gli ordini IOC è determinata dal trade server.	
SYMBOL_FILLING_MODE	Flags di di ordine consentito modalità di riempimento	SymbolInfoInteger
SYMBOL_ISIN	Il nome di un simbolo nel sistema ISIN (International Securities Identification Number). L'Interna	SymbolInfoString

Constant	Descrizione	Usage
	tional Securities Identification Number è un codice a 12 cifre alfanumerico che identifica in modo univoco una security. La presenza di questa proprietà del simbolo è determinata dal lato del trade server.	
SYMBOL_LAST	Prezzo dell'ultima transazione	SymbolInfoDouble
SYMBOL_LASTHIGH	Ultima massima del giorno	SymbolInfoDouble
SYMBOL_LASTLOW	Ultima minima del giorno	SymbolInfoDouble

Constant	Descrizione	Usage
SYMBOL_MARGIN_INITIAL	Il margine iniziale indica l'importo nella valuta di margine richiesto per aprire una posizione con il volume di un lotto. E' usato per controllare le attività di un cliente quando lui o lei entra nel mercato.	SymbolInfoDouble
SYMBOL_MARGIN_MAINTENANCE	Il margine di mantenimento. Se è impostato, imposta la quantità margine nella valuta margine del simbolo, caricato	SymbolInfoDouble

Constant	Descrizione	Usage
	di un lotto. E' usato per controllare le attività di un cliente quando cambiano gli status del suo account. Se il margine di mantenimento è uguale a 0, viene utilizzato il margine iniziale.	
SYMBOL_OPTION_MODE	Option type	SymbolInfoInteger
SYMBOL_OPTION_MODE_AMERICAN	American option may be exercised on any trading day on or before expiry. The period within which a buyer can exercise the	SymbolInfoInteger

Constant	Descrizione	Usage
	option is specified for it	
SYMBOL_OPTION_MODE_EUROPEAN	European option may only be exercised on a specified date (expiration, execution date, delivery date)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT	Option right (Call/Put)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_CALL	A call option gives you the right to buy an asset at a specified price	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_PUT	A put option gives you the right to sell an asset at a specified price	SymbolInfoInteger
SYMBOL_OPTION_STRIKE	The strike price of an	SymbolInfoDouble

Constant	Descrizione	Usage
	option. The price at which an option buyer can buy (in a Call option) or sell (in a Put option) the underlying asset, and the option seller is obliged to sell or buy the appropriate amount of the underlying asset	
SYMBOL_ORDER_LIMIT	Gli ordini Limit sono ammessi (Buy Limit e Sell Limit)	SymbolInfoInteger
SYMBOL_ORDER_MARKET	Gli ordini di Mercato sono consentiti (Buy e Sell)	SymbolInfoInteger
SYMBOL_ORDER_MODE	Flags dei consenti	SymbolInfoInteger

Constant	Descrizione	Usage
	ti tipi di ordine	
SYMBOL_ORDER_SL	Stop Loss è consentito	SymbolInfoInteger
SYMBOL_ORDER_STOP	Ordini di Stop sono ammessi (Buy Stop e Sell Stop)	SymbolInfoInteger
SYMBOL_ORDER_STOP_LIMIT	Ordini Stop-limit sono ammessi (Buy Stop Limit e Sell Stop Limit)	SymbolInfoInteger
SYMBOL_ORDER_TP	Take Profit è consentito	SymbolInfoInteger
SYMBOL_PATH	Percorso nell'albero del simbolo	SymbolInfoString
SYMBOL_POINT	Valore punti del Symbol	SymbolInfoDouble
SYMBOL_SELECT	Symbol è selezionato in Market Watch	SymbolInfoInteger
SYMBOL_SESSION_AW	Prezzo medio ponderat	SymbolInfoDouble

Constant	Descrizione	Usage
	o della sessione corrente	
SYMBOL_SESSION_BUY_ORDERS	Numero di ordini Buy in questo momento	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Volume corrente di ordini Buy	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	Prezzo di chiusura della sessione corrente	SymbolInfoDouble
SYMBOL_SESSION_DEALS	Numero di affari nella sessione corrente	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	Sommatoria dell'interesse aperto	SymbolInfoDouble
SYMBOL_SESSION_OPEN	Prezzo di apertura della sessione corrente	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MAX	Prezzo massimo della sessione corrente	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	Prezzo minimo della	SymbolInfoDouble

Constant	Descrizione	Usage
	sessione corrente	
SYMBOL_SESSION_PRICE_SETTLEMENT	Prezzo di liquidazione della sessione corrente	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	Numero di ordini Sell in questo momento	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Corrente volume di ordini Sell	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	Sommatoria del turnover della sessione corrente	SymbolInfoDouble
SYMBOL_SESSION_VOLUME	Sommatoria del volume degli affari della sessione corrente	SymbolInfoDouble
SYMBOL_SPREAD	Valore differenziale in punti	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	Indicazione di uno spread fluttuante	SymbolInfoInteger

Constant	Descrizione	Usage
SYMBOL_START_TIME	Data di inizio simbolo di trade (di solito usato per i futures)	SymbolInfoInteger
SYMBOL_SWAP_LONG	Valore di swap Long	SymbolInfoDouble
SYMBOL_SWAP_MODE	Modalità di calcolo swap	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Gli swap vengono caricati in denaro nella valuta di deposito del cliente	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Gli swap vengono caricati in denaro nella valuta margine del simbolo	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Gli swap vengono caricati in denaro nella valuta di base del simbolo	SymbolInfoInteger

Constant	Descrizione	Usage
SYMBOL_SWAP_MODE_DISABLED	Swap disabilitato (senza swap)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Gli swap vengono caricati come interesse specifico annuale del prezzo di uno strumento al calcolo dello swap (l'anno standard della banca è 360 giorni)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_OPEN	Gli swap vengono caricati come interesse specifico annuale del prezzo di apertura della posizione (l'anno standard della banca è	SymbolInfoInteger

Constant	Descrizione	Usage
	360 giorni)	
SYMBOL_SWAP_MODE_POINTS	Gli swap sono espressi in punti	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	Gli swap vengono caricati dalla riapertura delle posizioni. Al termine di una giornata di trading la posizione viene chiusa. Il giorno dopo viene riaperto dai punti +/- specificati del prezzo Bid corrente (parametri SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Gli swap vengono caricati dalla	SymbolInfoInteger

Constant	Descrizione	Usage
	riapertura delle posizioni. Al termine di una giornata di trading la posizione viene chiusa. Il giorno dopo viene riaperto dai punti +/- specificati del prezzo di chiusura (parametri SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_ROLLOVER3DAYS	Giorno della settimana per caricare swap di rollover a 3 giorni	SymbolInfoInteger
SYMBOL_SWAP_SHORT	Valore di swap Short	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	Numero massimo	SymbolInfoInteger

Constant	Descrizione	Usage
	di richieste mostrate in Profondità di Mercato. Per i simboli che non hanno coda di richieste, il valore è uguale a zero.	
SYMBOL_TIME	Orario dell'ultima quotazione	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	Modalità di calcolo del prezzo del contratto	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	Grandezza del contratto di trade	SymbolInfoDouble
SYMBOL_TRADE_EXECUTION_EXCHANGE	Esecuzione di scambio	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	Esecuzione istantanea	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	Esecuzione a	SymbolInfoInteger

Constant	Descrizione	Usage
	mercato	
SYMBOL_TRADE_EXECUTION_REQUEST	Esecuzione a richiesta	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	Modalità di esecuzione affare	SymbolInfoInteger
SYMBOL_TRADE_FREEZE_LEVEL	Distanza per congelare le operazioni di trade in punti	SymbolInfoInteger
SYMBOL_TRADE_MODE	Tipo ordine di esecuzione	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	Permesso solo operazioni di chiusura posizioni	SymbolInfoInteger
SYMBOL_TRADE_MODE_DISABLED	Il trade è disabilitato per il simbolo	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	Non ci sono restrizioni trade	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	Permesso solo per posizioni long	SymbolInfoInteger

Constant	Descrizione	Usage
SYMBOL_TRADE_MODE_SHORTONLY	Permesso solo per posizioni short	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	Rientro minimo in punti dal prezzo corrente più vicino per piazzare ordini di Stop	SymbolInfoInteger
SYMBOL_TRADE_TICK_SIZE	Cambiamento prezzo minimo	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	Valore di SYMBOL_TRADE_TICK_VALUE_PROFIT	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_LOSS	Prezzo tick calcolato per una posizione in perdita	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_PROFIT	Prezzo tick calcolato per una posizione favorevole	SymbolInfoDouble
SYMBOL_VOLUME	Volume dell'ultim	SymbolInfoInteger

Constant	Descrizione	Usage
	a transazione	
SYMBOL_VOLUME_LIMIT	Massimo volume consentito aggregato di una posizione aperta ed ordini pendenti in una direzione (acquisto o vendita) per il simbolo. Per esempio, con la limitazione di 5 lotti, si può avere una posizione aperta buy con il volume di 5 lotti e posizionare un ordine pendente Sell Limit con il volume di 5 lotti. Ma	SymbolInfoDouble

Constant	Descrizione	Usage
	in questo caso non è possibile inserire un ordine pendente Buy Limit (in quanto il volume totale in una direzione supererà il limite) o piazzare Sell Limit con il volume più di 5 lotti.	
SYMBOL_VOLUME_MAX	Volume massimo per un affare	SymbolInfoDouble
SYMBOL_VOLUME_MIN	Volume minimo per un affare	SymbolInfoDouble
SYMBOL_VOLUME_STEP	Cambio di step minimo per l'esecuzione dell'affare	SymbolInfoDouble
SYMBOL_VOLUMEHIGH	Volume massimo	SymbolInfoInteger

Constant	Descrizione	Usage
	del giorno	
SYMBOL_VOLUMELOW	Volume minimo del giorno	SymbolInfoInteger
TENKANSEN_LINE	Linea Tenkan-sen	Linee Indicatori
TERMINAL_BUILD	Il numero di build del terminale client	TerminalInfoInteger
TERMINAL_CODEPAGE	Numero della pagina codice della lingua installati nel terminale client	TerminalInfoInteger
TERMINAL_COMMONDATA_PATH	Percorso comune per tutti i terminali installati in un computer	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	Account in MQL5.community	TerminalInfoInteger
TERMINAL_COMMUNITY_BALANCE	Bilancio in MQL5.community	TerminalInfoDouble

Constant	Descrizione	Usage
TERMINAL_COMMUNITY_CONNECTION	Collegamento a MQL5.community	TerminalInfoInteger
TERMINAL_COMPANY	Nome della società	TerminalInfoString
TERMINAL_CONNECTED	La connessione ad un trade server	TerminalInfoInteger
TERMINAL_CPU_CORES	Il numero di CPU cores nel sistema	TerminalInfoInteger
TERMINAL_DATA_PATH	Cartella in cui sono memorizzati i dati dei terminali	TerminalInfoString
TERMINAL_DISK_SPACE	Spazio libero su disco per la cartella MQL5\Files del terminale(agente), MB	TerminalInfoInteger
TERMINAL_DLLS_ALLOWED	Il permesso di utilizzare DLL	TerminalInfoInteger
TERMINAL_EMAIL_ENABLED	Il permesso di	TerminalInfoInteger

Constant	Descrizione	Usage
	inviare e-mail utilizzando l' SMTP-server ed il login, specificato nelle impostazioni del terminale	
TERMINAL_FTP_ENABLED	Il permesso di inviare i reports con server FTP ed il login, specificato nelle impostazioni del terminale	TerminalInfoInteger
TERMINAL_LANGUAGE	Lingua del terminale	TerminalInfoString
TERMINAL_MAXBARS	Il massimo conteggio di barre nel grafico	TerminalInfoInteger
TERMINAL_MEMORY_AVAILABLE	Memoria libera del processo del terminal	TerminalInfoInteger

Constant	Descrizione	Usage
	e(agente), MB	
TERMINAL_MEMORY_PHYSICAL	Memoria fisica nel sistema, MB	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	Memoria disponibile per il processo del terminale(agente), MB	TerminalInfoInteger
TERMINAL_MEMORY_USED	Memoria utilizzata dal terminale(agente), MB	TerminalInfoInteger
TERMINAL_MQID	The flag indicates the presence of MetaQuotes ID data for Push notifications	TerminalInfoInteger
TERMINAL_NAME	Nome del terminale	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	Permesso di inviare notifiche smartphone	TerminalInfoInteger

Constant	Descrizione	Usage
TERMINAL_OPENCL_SUPPORT	La versione del OpenCL supportato nel formato di 0x00010002 = 1.2. "0" significa che OpenCL non è supportato	TerminalInfoInteger
TERMINAL_PATH	Cartella da cui viene avviato il terminale	TerminalInfoString
TERMINAL_PING_LAST	L'ultimo valore conosciuto del ping al trade server in millisecondi. Un secondo comprende un milione di microsecondi.	TerminalInfoInteger
TERMINAL_SCREEN_DPI	La risoluzione del display informativo sullo	TerminalInfoInteger

Constant	Descrizione	Usage
	schermo è misurata come ammontare di Punti in una linea per Inch (DPI)	
TERMINAL_TRADE_ALLOWED	Permesso per il trade	TerminalInfoInteger
TERMINAL_X64	Indicazione del "terminale 64-bit"	TerminalInfoInteger
THURSDAY	Giovedi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	Piazza un ordine di trade per l'immediata esecuzione con i parametri specificati(market order)	MqlTradeRequest
TRADE_ACTION_MODIFY	Modifica i parametri dell'ordine piazzato precede	MqlTradeRequest

Constant	Descrizione	Usage
	ntemente	
TRADE_ACTION_PENDING	Posiziona un ordine di trade per l'esecuzione sotto condizioni specificate (ordine pendente)	MqlTradeRequest
TRADE_ACTION_REMOVE	Eliminare l'ordine pendente piazzato precedentemente	MqlTradeRequest
TRADE_ACTION_SLTP	Modifica i valori Stop Loss e Take Profit per una posizione aperta	MqlTradeRequest
TRADE_RETCODE_CANCEL	Richiesta annullata dal trader	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	Autotrading disabilitato dal	MqlTradeResult

Constant	Descrizione	Usage
	terminale client	
TRADE_RETCODE_CONNECTION	Nessuna connessione con il trade server	MqlTradeResult
TRADE_RETCODE_DONE	Richiesta completata	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	Solo una parte della richiesta è stata completata	MqlTradeResult
TRADE_RETCODE_ERROR	Errore elaborazione richiesta	MqlTradeResult
TRADE_RETCODE_FROZEN	Ordine o posizione congelati	MqlTradeResult
TRADE_RETCODE_INVALID	Richiesta non valida	MqlTradeResult
TRADE_RETCODE_INVALID_EXPIRATION	Data di espirazione dell'ordine non valida nella richiesta	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	Non valido tipo di filling	MqlTradeResult

Constant	Descrizione	Usage
	dell'ordine	
TRADE_RETCODE_INVALID_ORDER	Errato o proibito tipo di ordine	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	Prezzo non valido nella richiesta	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	Stops non validi nella richiesta	MqlTradeResult
TRADE_RETCODE_INVALID_VOLUME	Volume non valido nella richiesta	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	Il numero di ordini in corso ha raggiunto il limite	MqlTradeResult
TRADE_RETCODE_LIMIT_VOLUME	Il volume degli ordini e posizioni per il simbolo ha raggiunto il limite	MqlTradeResult
TRADE_RETCODE_LOCKED	Richiesta bloccata	MqlTradeResult

Constant	Descrizione	Usage
	pe l'elaborazione	
TRADE_RETCODE_MARKET_CLOSED	Il mercato è chiuso	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	Nessuna modifica nella richiesta	MqlTradeResult
TRADE_RETCODE_NO_MONEY	Non ci sono abbastanza soldi per completare la richiesta	MqlTradeResult
TRADE_RETCODE_ONLY_REAL	Il funzionamento è consentito solo per i conti live	MqlTradeResult
TRADE_RETCODE_ORDER_CHANGED	Stato ordine cambiato	MqlTradeResult
TRADE_RETCODE_PLACED	Ordine piazzato	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	La posizione con lo specificato POSITION_IDENTIFIER è stata già chiusa	MqlTradeResult

Constant	Descrizione	Usage
TRADE_RETCODE_PRICE_CHANGED	Prezzi cambiato	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	Non ci sono quotazioni per elaborare la richiesta	MqlTradeResult
TRADE_RETCODE_REJECT	Richiesta rigettata	MqlTradeResult
TRADE_RETCODE_REQUOTE	Riquotazione	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	Autotrading disabilitato dal server	MqlTradeResult
TRADE_RETCODE_TIMEOUT	Richiesta annullata per timeout	MqlTradeResult
TRADE_RETCODE_TOO_MANY_REQUESTS	Richieste troppo frequenti	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	Il trade è disattivato	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	Aggiunta di un affare per la cronistoria. L'azione viene eseguita come	MqlTradeTransaction

Constant	Descrizione	Usage
	risultato di un' esecuzione di ordine o l'esecuzione di operazioni su un saldo del conto.	
TRADE_TRANSACTION_DEAL_DELETE	Eliminazione di un affare dalla cronistoria. Ci possono essere casi in cui un accordo precedentemente eseguito viene eliminato da un server. Per esempio, un accordo è stato eliminato in un sistema di trade esterno (scambio) in cui era precedentemente	MqlTradeTransaction

Constant	Descrizione	Usage
	trasferito da un broker.	
TRADE_TRANSACTION_DEAL_UPDATE	Aggiornamento di un affare nella cronistoria. Ci possono essere casi in cui un accordo applicato in precedenza viene modificato su un server. Per esempio, un accordo è stato cambiato in un sistema commerciale esterno (scambio) in cui era precedentemente trasferito da un broker.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_ADD	Aggiunta di un ordine	MqlTradeTransaction

Constant	Descrizione	Usage
	allo storico come risultato di esecuzione o cancellazione.	
TRADE_TRANSACTION_HISTORY_DELETE	Deleting an order from the orders history. Questo tipo viene fornito per migliorare la funzionalità sul lato trade server.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_UPDATE	Modifica un ordine situato nella storia ordini. Questo tipo viene fornito per migliorare la funzionalità sul lato trade server.	MqlTradeTransaction

Constant	Descrizione	Usage
TRADE_TRANSACTION_ORDER_ADD	L'aggiunta di un nuovo ordine aperto.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	La rimozione di un ordine dalla lista di quelli aperti. Un ordine può essere eliminato da quelli aperti a seguito dell'impostazione di un'appropriata richiesta o esecuzione (riempimento) e spostandosi nella cronistoria.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_UPDATE	Aggiornamento di un ordine aperto. Gli aggiorna	MqlTradeTransaction

Constant	Descrizione	Usage
	menti includono non solo le evidenti modifiche dal terminale client o dal lato trade server, ma anche cambiamenti dello stato di un ordine quando lo si imposta (ad esempio, passaggio da ORDER_STATE_STARTED a ORDER_STATE_PLACED o da ORDER_STATE_PLACED a ORDER_STATE_PARTIAL, ecc).	
TRADE_TRANSACTION_POSITION	Modifica di una posizion	MqlTradeTransaction

Constant	Descrizione	Usage
	e non correlata ad un'esecuzione di affare. Questo tipo di transazione mostra che una posizione è stata modificata dal lato trade server. Volume della posizione, prezzo di apertura, Stop Loss e Take Profit possono essere modificati. I dati sulle modifiche vengono inviati nella struttura MqlTradeTransaction tramite l'handler OnTradeTransact	

Constant	Descrizione	Usage
	ion. La modifica della posizione (aggiunta, modifica o chiusura), come risultato dell'esecuzione di un affare, non porta al verificarsi della transazione TRADE_TRANSACTION_POSITION.	
TRADE_TRANSACTION_REQUEST	La notifica del fatto che una richiesta di trade sia stata elaborata dal server ed il risultato dell'elaborazione sia stato ricevuto. Solo il campo tipo (tipo di	MqlTradeTransaction

Constant	Descrizione	Usage
	transazione di trade) deve essere analizzato per tali operazioni nella struttura MqlTradeTransaction. Il secondo e terzo parametro di OnTradeTransaction (richiesta e risultato) devono essere analizzati per ulteriori dati.	
TUESDAY	Martedì	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	color	MqlParam
TYPE_DATETIME	datetime	MqlParam
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam
TYPE_INT	int	MqlParam
TYPE_LONG	long	MqlParam

Constant	Descrizione	Usage
TYPE_SHORT	short	MqlParam
TYPE_STRING	string	MqlParam
TYPE_UCHAR	uchar	MqlParam
TYPE_UINT	uint	MqlParam
TYPE_ULONG	ulong	MqlParam
TYPE_USHORT	ushort	MqlParam
UCHAR_MAX	Valore massimo, che può essere rappresentato dal tipo uchar	Costanti di tipo numerico
UINT_MAX	Valore massimo, che può essere rappresentato dal tipo unit	Costanti di tipo numerico
ULONG_MAX	Valore massimo, che può essere rappresentato dal tipo ulong	Costanti di tipo numerico
UPPER_BAND	Limite superiore	Linee Indicatori
UPPER_HISTOGRAM	Istogramma superiore	Linee Indicatori
UPPER_LINE	Linea superior	Linee Indicatori

Constant	Descrizione	Usage
	e	
USHORT_MAX	Valore massimo, che può essere rappresentato dal tipo ushort	Costanti di tipo numerico
VOLUME_REAL	Trade volume	Costanti Prezzo
VOLUME_TICK	Tick volume	Costanti Prezzo
WEDNESDAY	Mercoledì	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	Indica il numero di elementi restanti fino alla fine dell'array, cioè, verrà elabolrato l'intero array	Altre costanti
WRONG_VALUE	La costante può essere implicitamente lanciata a qualsiasi tipo di enumerazione	Altre costanti